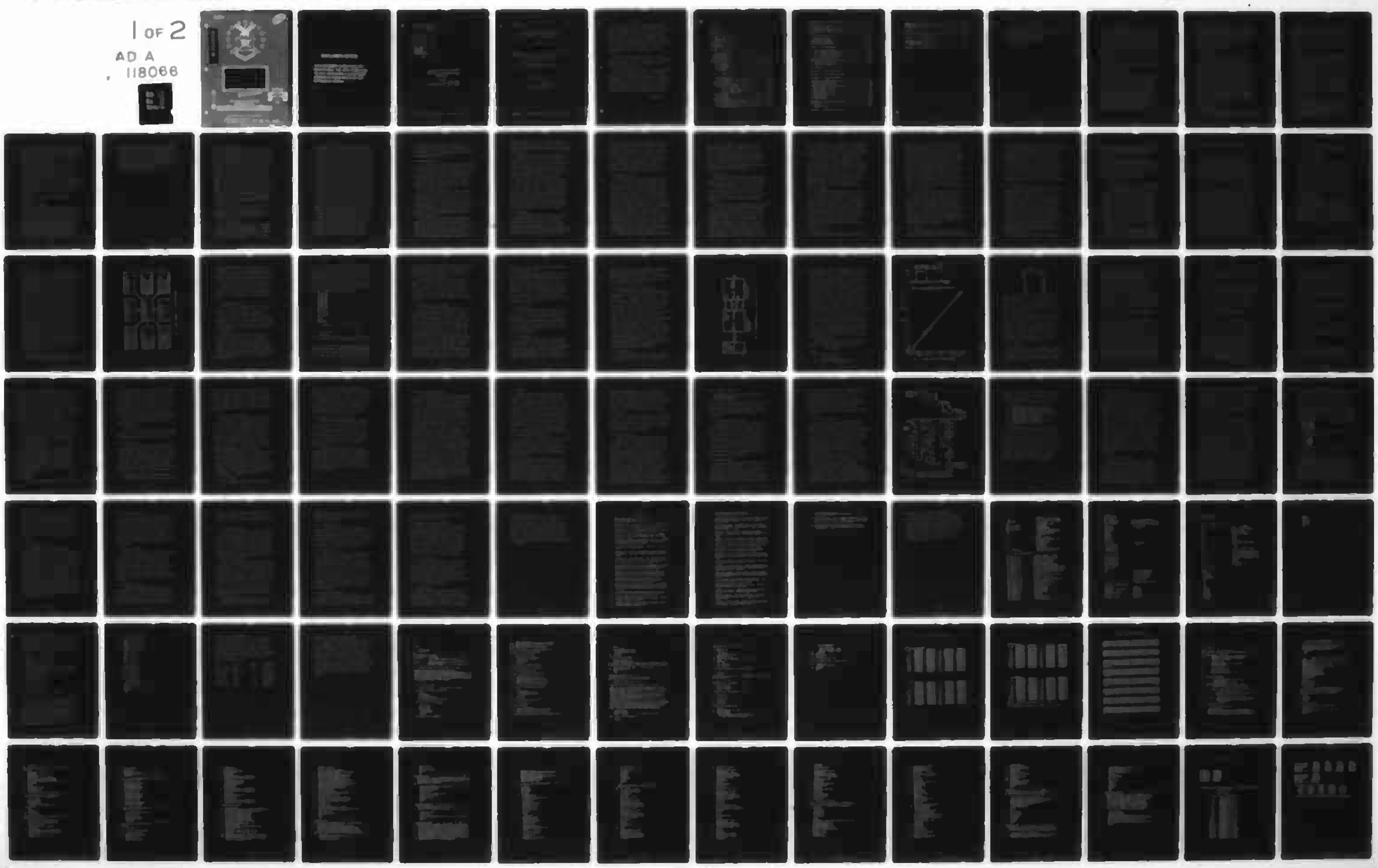
AD-A118 066 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 17/2
A GENERAL PURPOSE MINI-COMPUTER BASED DIGITAL SIGNAL PROCESSING--ETC(U)
JUN 82 W TODD
UNCLASSIFIED AFIT/GE/EE/82J-14 NL

1 OF 2
AD A
118066



A

O

ODC

1

AD A118066

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY (ATC)

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

82 08 11 063

DISCLAIMER NOTICE

AFIT/GE/EE/82J-14

A GENERAL PURPOSE MINI-COMPUTER BASED DIGITAL SIGNAL PROCESSING LABORATORY

THESIS

AFIT/GE/EE/82J-14 WAYNE TODD 2LT USAF

AFIT/GE/EE/82J-14

A GENERAL PURPOSE MINI-CCMPUTER BASED DIGITAL SIGNAL PROCESSING LABORATORY

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

by

Wayne Todd, B.S.
2Lt USAF
Graduate Electrical Engineering

June 1982

Approved for public release; distribution unlimited.

Preface

The Air Force has seen a growing number of weapon systems that employ some type of digital signal processing (DSP). Many analog circuits are being replaced with digital equivalent circuits. AFIT supports this trend with courses, laboratory projects, and theses designed to give students sufficient background in this area to satisfy Air Force needs. The computer facilities in the AFIT engineering laboratory are an important part of this support.

One of the computer systems has recently had more hardware added to it, making it potentially a very valuable resource in support of DSP. This report describes what was accomplished with the system.

Thanks to Professor Gary Lamont, advisor, who proposed the topic and helped limit its scope. His interest, encouragement, and suggestions were invaluable. Thanks to John Bankovskis, currently assigned to the Optical Sensors Lab, who volunteered his time and expertise to assist in the generation of the operating system. Thanks also to Maj Lillie and Capt Kizer, my thesis-committee members.

Wayne Todd

Table of Contents

List of Figures

Abstract

This project involved the integration of hardware and software to develop a general purpose digital signal processing (DSP) laboratory based on a Hewlett Packard (HP) 21MX-type computer. The hardware included an HP 2108A computer, HP 7906 disk drive, HP 2648A graphics terminal, Texas Instruments Silent 700 printing terminal, Remex optical paper tape reader, two Analog Devices ADC-12QZ analog-to-digital converters, and an HP 12555B digital-to-analog converter.

The operating system generated for the laboratory was RTE-III. It is the latest version of RTE that can be used with the currently available hardware. Software support for I/O devices not supported by RTE-III was developed and implemented.

Four DSP applications programs were implemented and tested. The first program performs both a fast Fourier transform (FFT) and an inverse fast Fourier transform (IFFT) of a complex function. The FFT and IFFT are computationally efficient methods for time-to-frequency and frequency-to-time transformations, respectively. The second program designs a linear phase finite impulse response digital filter. The third program uses auto-correlation and covariance methods of linear prediction analysis. The fourth program designs a finite word-length infinite impluse response digital filter.

Chapter 1 Introduction

This chapter begins with background information on the motivation supporting this study. This is followed by a brief look at the requirements for a general purpose digital signal processing (DSP) laboratory, the extent to which these requirements have been met, and the approach taken. The chapter concludes with an introductory description of the thesis project.

1.1 Background

The AFIT engineering laboratory has had an HP 21MX computer system for several years. It has not been used extensively by either students or faculty because its limited peripheral devices and limited software support made its use difficult. Recently, additional hardware (e.g., disk drive, graphics terminal, memory) has been added to the system which would potentially make it a very valuable asset to AFIT engineering students and faculty. These additional capabilities have not previously been exploited. Also, professor Gary Lamont, who is responsible for the system's use, has designated it to function as a DSP laboratory. DSP is an important part of the engineering curriculum, and additional support facilities are needed to meet the demands of a growing number of students.

1.2 Requirements

The basic requirement of the laboratory is that it be user-oriented and provide hardware and software to

support most common DSP applications. It should also be expandable to allow the integration of additional hardware and software. The basic requirement has been divided into six requirements for the purpose of discussion and implementation. These are: 1) the operating system, 2) data acquisition, 3) data storage, 4) computational tasks, 5) output display, and 6) user interaction.

1.3 Scope

This project investigated and demonstrated most of the system's capabilities as they existed at the beginning, and as they were added during the project. All the hardware had been specified and ordered previously. Software has been provided (including an operating system) where necessary in support of the hardware. Also, several DSP applications programs were taken from the literature and modified to run on the HP system. Investigation of the present capabilities led to conclusions about what additions should be made to meet the ultimate design goal. The recommended additions are presented as either essential or desireable.

1.4 Approach

The first task was to investigate the requirements for a general purpose DSP laboratory. This involved both a literature search and personal conversations with several people currently using DSP. The next task was to generate an operating system (OS). The OS provides

software support for most of the I/O devices used. It also has program development and testing facilities that were necessary later in the project. The OS was generated on another HP computer system because a disk loader ROM, which is necessary to load the OS into main memory, was not available early in the project.

Next, the hardware was configured. Most of it was available from the start, but some (e.g., the disk loader ROM) arrived later. RTE-III supports most of the hardware, but software support had to be provided for the A/D converters and the D/A converter. Four methods of software support for the converters were investigated: polling, interrupts, direct memory access, and system I/O drivers. This development followed the hardware integration.

Finally, several DSP applications programs were implemented and tested.

1.5 Overview of Thesis

Chapter 2 covers the requirements that have been identified for the laboratory. The requirements are a combination of observing similar systems and the author's own judgement. Chapter 3 discusses the system's hardware and support software, which together comprise the laboratory's environment. The main subject areas covered are the operating system, data acquisition, data storage, and the output display. Chapter 4 is a discussion of the DSP applications programs that have

been implemented and tested. The four programs presented were taken from a textbook (Ref 7) and modified to run on the HP system. Also presented, is a possible way to use the system for real-time processing. Chapter 5 is a discussion of the issues which have been addressed pertaining to user interaction. Chapter 6 lists and justifies numerous recommendations which would allow the DSP laboratory to meet all requirements.

Chapter 2 Requirements

A literature search did not lead to any book, periodical, or other document which specified and justified either the hardware or software requirements for a general purpose DSP laboratory. However, existing hardware and software implementations were studied (Ref 4:433-441, 7, 10, 12, 18, 23, 25:655-657, 31, 32, 33, Appendix C) in an attempt to determine the most common requirements, and survey possible capabilities. Six basic requirement areas have been identified: the operating system, data acquisition, data storage, computational tasks, output display, and user interaction. Even though each of these areas is discussed in a separate section of this chapter, they are closely interrelated and must be considered together when implementing a DSP laboratory. Note that for brevity, in this context, the term operating system includes the assembler, interactive editor, compilers, and system utilities.

The overall objective is to design a user-oriented system so that the DSP system user does not require a computer background. This objective is to be fullfilled in both an educational and research environment.

2.1 Operating System

Since the user of this DSP laboratory is assumed to have minimum knowledge of the specifics of computer operations, automated resource management is essential. The operating system (OS) provides this. In effect,

the details of resource management should be "user-transparent" (Ref 20). These resources include the CPU, memory, peripheral devices, and files. Because the OS manages these resources, it is an important part of user interaction, discussed separately in section 2.6.

An OS can be classified as a real-time OS, batch OS, time-sharing OS, or multipurpose OS (Ref 19:61-99). A real-time OS is characterized as being able to perform a sequence of processing tasks (e.g., real-time data acquisition, computations, and display) within rigid time constraints.A batch OS is capable of executing more than one job without the need for operator intervention. The concept of time-sharing refers to distributed computer power among a set of users at interactive terminals. A multipurpose OS will support the capabilities of a combination of at least two of the other three types of OSs.

The DSP laboratory requires a multipurpose OS. It is intended to have the capability for real-time processing as described in section 4.5. Also, many DSP programs have lengthy (one or more hours) execution times. Therefore, a multiprogramming batch capability is also required, so that the user can initiate actions while one or more programs are executing. The only aspect of time-sharing required is the use of an interactive terminal. This is because the system is being designed such that there can only be one user at a time. An interactive terminal is more user-responsive than a

batch system because it allows on-line inputs to be entered by the user. Finally, as an additional aid in user-responsiveness, the user should be able to initiate batch programs interactively.

2.2 Data Acquisition

Data can be acquired through a program that generates data, from a data file, or from an external source. In the case of an external source, the phenomena is measured by some device and is represented as an analog signal. An analog-to-digital (A/D) converter will convert the analog signal (samples taken in real-time and equally spaced in time) into a form (binary numbers) suitable for digital computer processing.

Several problem areas exist with regard to representing analog data in digital form. The first is the accuracy and precision of the A/D converter. The precision is determined by the number of bits which represent each data point. The accuracy concerns the ability of the A/D converter to provide correct data within the precision limit. Two other common problem areas are aliasing and leakage (Ref 2).

Aliasing refers to the fact that high-frequency components of a time function can impersonate low frequencies if the sampling rate is too low. To effectively deal with aliasing, the sampling frequency must be at least twice that of the highest frequency, W present in the signal being analyzed. However, increasing the

sampling frequency results in more data points and consequently, more computations. For band limited signals, Chen (Ref 4:82-83) recommends a sampling frequency between 2.0W and 2.5W.

The problem of leakage arises from sampling a signal for a finite period and neglecting what happened before or after this period. This is equivalent to multiplying the signal by a rectangular data window, which results in the leakage of energy from one discrete frequency into adjacent frequencies. Various types of data windows can be used to reduce leakage (Ref 24).

The user should be able to interactively specify the sampling rate, the number of samples per data frame, the type of data window (if any), and the data source (e.g., an I/O channel number). By changing the first three of these four variables and analyzing the same data several times, error analysis can be performed with respect to aliasing and leakage. However, except for periodic calibration, the user has to assume that the data supplied by the A/D converter is correct.

2.3 Data Storage

Data acquired from an external source or generated by a program is initially stored in main memory. In the case of the external source, it is because main memory is faster than secondary memory. The data can then be sent to a display device, be used in computations, or stored on a mass storage device, such as a disk.

The capability for storing data is especially important because some data is relatively expensive or difficult to obtain. For example, it may be expensive to turn on a rocket engine for the purpose of obtaining sensor data. This kind of test may also be difficult to set up.

Another reason for storing data is that it may be desired to use the same data more than once. For example, it can be recalled and processed by one algorithm or several algorithms, and results compared.

2.4 Computational Tasks

The computational tasks, or algorithms presented here are for a general purpose DSP laboratory. However, specific applications may require additional types of computation, or analysis of several methods of accomplishing the same task. Therefore, the overall software structure should be modular, so that additional modules can be added without the need to alter the existing software. A module is a separate entity of instructions, which by itself performs a predefined task, computational or otherwise. Computational tasks required for a general purpose DSP laboratory include the fast Fourier transform (FFT), inverse FFT (IFFT), transfer function, coherence function, auto correlation, cross correlation, convolution, waveform averaging, various statistical computations, digital filter design, different types of data windows, and power spectral density (Ref 32).

The system should have the capability of performing

these tasks in real-time, if appropriate (e.g., FFT, IFFT, correlation, etc.). To insure this, the CPU, memory, and executing programs must be fast enough to allow real-time computations without the loss of data (i.e., losing data by replacing old data with new data before the old data has been used). In this case, computations must be performed in parallel with real-time data acquisition and real-time display.

2.5 Output Display

A general purpose DSP laboratory requires the capability for displaying data and results from the aforementioned computational tasks. Also, a display device (e.g., a graphics terminal) is an essential communication medium between the user and the computer in an interactive environment. However, a discussion of this second aspect of the output display is reserved for section 2.6.

Requirements for the output display must be specified both in terms of the types of devices and the format of the display. Several criteria exist for selecting the display devices. First, the system should provide the user with a hard copy of any useful display, for documentation purposes. This might include a line printer for listing tables of values, or a camera for taking a picture of an oscilloscope display. A soft copy device (CRT) is necessary for some types of real-time display where the display is continuously replaced

with a new display. It is also important to consider whether the soft copy device is a storage or non-storage type. A storage device does not require continuous input from the computer for a display, whereas a non-storage device does.

Rather than stating that one or more particular devices be required, the combined capability of a particular group of devices should be considered in light of the relevant issues presented. In addition, some applications may require an output which is not a visual display. For example, an audio amplifier and speaker are needed for speech synthesis. A general purpose DSP laboratory will not be able to support all applications, but it should be flexible enough to allow additional output devices to be added.

The information displayed should be in a format which is convenient and comprehensible to the user. Excluding the type of display device, the type of output format should be governed primarily by the information the user hopes to extract from the output. For example, a graph generally provides a good panoramic view of a set of data, whereas a list of values can provide more information (e.g., number of significant digits) about specific data points. The user should be able to interactively select from various formats to suit his particular needs.

2.6 User Interaction

Generally speaking, the computer cannot serve as a

useful tool unless a good system of communication has been established between the user and the computer. The purpose of this section is to present the major issues involved in designing a good communication link. Both psychological and technical aspects should be considered, as with any other human engineering problem (Ref 9:2). Basically, the computer must be adapted to the user. The OS, as previously discussed, is an important first step in providing a degree of transparency for the user. However, this only applied to system resources. A higher level of transparency is required for application programs (e.g., DSP) so that they will also be user-oriented. Higher level just means that applications programs are scheduled by the OS.

The issues concerning user interaction presented are: the user, response time, language, input, and user assistance. Even a system which addresses all of these issues with some degree of success should make provision for user feedback and subsequent changes (Ref 22:61). The system designer should either take the responsibility for this or provide sufficient documentation for the system manager to take appropriate action. With adherence to these criteria, the ultimate goal of user satisfaction can be attained.

2.6.1 The User

In order to design a DSP laboratory for user satisfaction, the user must be defined. As stated previously,

the users will be students and faculty members at AFIT. This presents the problem of designing a system for a group of users with diverse backgrounds (Ref 29:17). One requirement is that the user should not need a computer background. Users will also have varying objectives when using the laboratory. The spectrum of objectives can be defined as ranging from simply entering commands and accompanying parameters for performing DSP experiments, to modification and upgrade of the system's capabilities. The issues of user interaction which are addressed here pertain to users with the former objective.

2.6.2 Response Time

User satisfaction regarding response time after a command has been entered depends both on the user's expectations and the system's capabilities (Ref 22:73). Expectations can be modified initially by providing approximate execution time values in the documentation. Of course, expectations will also change through experience in using the laboratory. This section will focus only on design issues pertinent to minimizing response time. Response time in this context is defined as that time between the user initiating an action (e.g., retrieval of a data file, computations) to the system's completion of that action (if possible), and notifying the user that it is completed or that it cannot be completed.

As stated previously, the system is intended to

support only one interactive user at a time. Therefore, the problems associated with minimizing response time for multiple users in a time-sharing environment are not discussed.

With this constraint, the focus of attention is on speed of execution. The speed depends on the hardware, software (including the OS), and the algorithms employed. Issues pertinent to execution speed are far too extensive to adequately cover in this report. The interested reader is referred to the literature.

2.6.3 Language

The language that the user uses to communicate with the system for the purpose of initiating DSP operations is called the command language (CL). The CL should resemble, as much as possible, human communication (Ref 9:27). This is because the design process has the goal of adapting the system to humans, rather than the reverse. Therefore, commands should be English words instead of codes. For example, the command "REPLACE" is much clearer than "RP". However, once the user gains familiarity with the system it is convenient to have the option of entering a shortened version of each command to reduce the number of input characters required.

The other half of designing for human communication involves the computer's responses to user requests. Both requests for input by the computer and notification of

the system's status should be in a readable English format.

2.6.4 Input

The discussion of user input in this section assumes an interactive environment, although most of the requirements also apply to batch. These requirements are: user interruption of a requested action, interception of user input errors by the system, correction of input errors by the user, provision of default values, logical sequencing of inputs, construction of a sequence of user requests, and useful error messages (Ref 17).

Once the user has caused the system to begin some action, he should have the means of interrupting the process. This feature is useful for recovering from user mistakes which are not unacceptable errors to the system. Those errors which are invalid (e.g., exceed parameters, illegal commands) should be in intercepted by the system before execution is attempted. The user should be notified at that point of the specific error and be given options for further action. One of the options should allow the user to correct the erroneous input. To aid in avoiding errors in the first place, default values should be provided, as appropriate. Another way to avoid input errors is for the system to request the input in a logical order. Finally, the user should be able to construct a sequence of

commands (commonly called a MACRO) so that he doesn't have to wait for each action to complete prior to entering the next command.

2.6.5 User Assistance

Both on-line and off-line assistance are necessary for a user-responsive system. On-line assistance includes system prompts for user input, lists of options, and summary information for each valid command. Prompts are required so that the user will always be aware of the nature of the input requested. A list of options should always follow an error message (Ref 22:112), and may also be used with prompts. Summary information, often called HELP files, should give the user a brief description of each valid command and the limits for each of its parameters (Ref 17).

Off-line assistance refers to the documentation provided for the DSP laboratory. A system reference manual should be included which contains comprehensive information regarding all of the system's capabilities not covered in the other system documents. A training manual should also be included. It should teach the user how to properly use every available command through the use of examples and exercises (Ref 22:56).

Chapter 3 Environment

The laboratory environment can be divided into four separate categories: the operating system (OS), data acquisition, data storage, and output display. These four together provide the capabilities and impose the ultimate limitations under which the user must work.

At the beginning of this project, the specific hardware environment included an HP 2108A computer, HP 7906 disk drive, HP 2648A graphics terminal, Remex high speed paper tape reader, teletype terminal with paper tape punch/reader, two A/D and one D/A converter, memory boards, and various I/O interfaces.

The graphics terminal was not initially configured because the interface board for it had not yet arrived. The disk drive couldn't be used at first because a loader ROM had not arrived. Without the disk drive, the operating system couldn't be generated for the desired configuration. However, all the hardware listed above (except the teletype) is now configured, with the arrival of the required components (Fig. 1). The teletype has been replaced with a Texas Instruments (TI) Silent 700 printing terminal. The OS has been generated to support all but the A/D and D/A converters. RTE-III does not include system drivers for these devices.

3.1 Operating System

RTE-III was chosen as the OS because it is the latest version of RTE that can operate with the currently

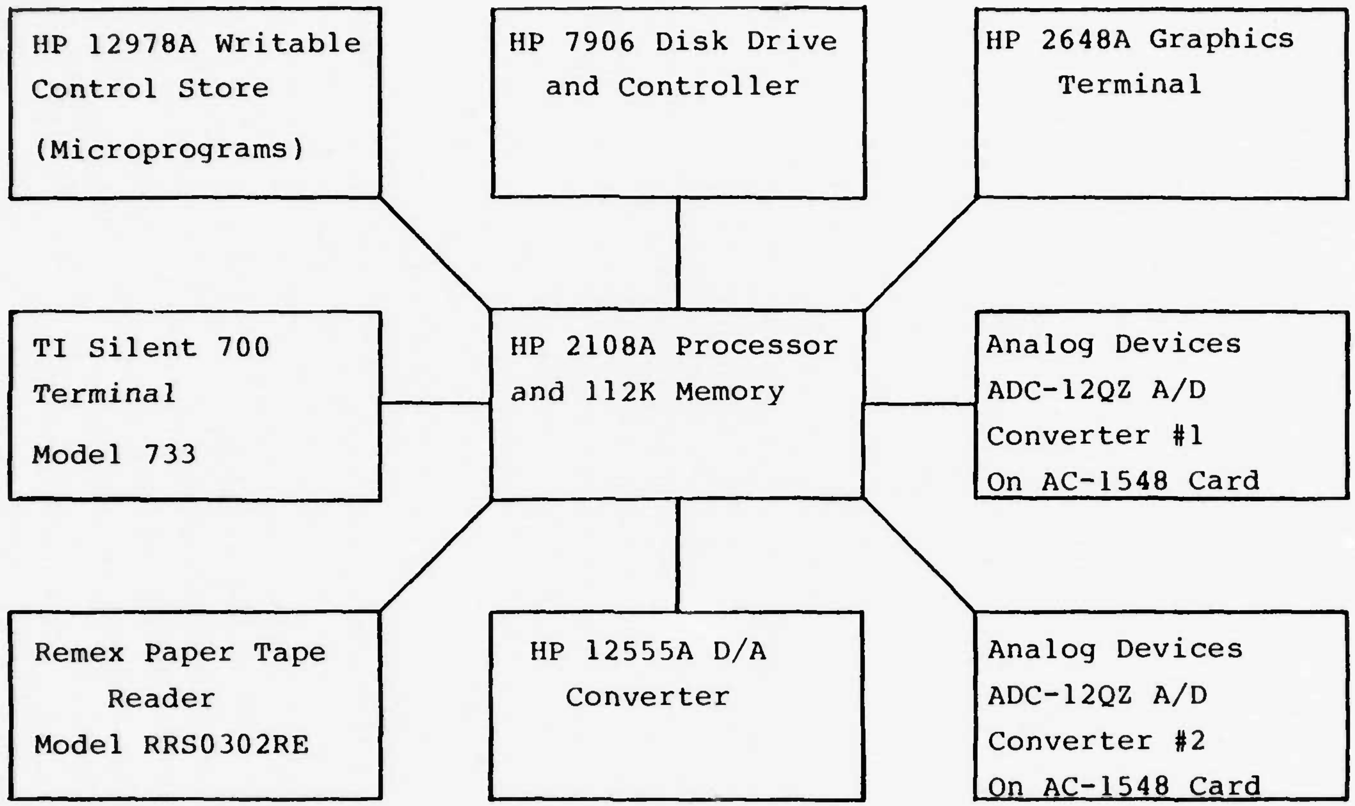


Fig. 1 Block Diagram of Configured Hardware

available support hardware (RTE stands for Real Time Executive) (Ref 26:1-1). The unique hardware required for RTE-III is the Dual Channel Port Controller (DCPC-used for direct memory access), dynamic memory mapping, memory protect, and a minimum of 32K words of memory.

First, the generation of RTE-III is discussed. This should assist future users who need to generate a new OS. Next, the capabilities and limitations of RTE-III are discussed with respect to the DSP application.

3.1.1 Generation of RTE-III

The procedures for generating a new OS begin by running a program called "RT3GN". The program, as it is running, will ask for various parameters to be entered from the system console. Since over a hundred responses are required, the preferred method of entering the parameters is to create an answer file. See Appendix A for a listing of the answer file used in the most recent generation.

Generating an OS takes about two hours if every entry in the answer file is valid. Unfortunately, the computer usually halts before the generation is complete. The point at which it halts appears to be random. Why this anomaly occurs is presently unknown. The current version of the OS required 50 to 60 attempts before it finally completed (using the same answer file every time).

Once the OS has been generated, a program called "SWTCH" is run to switch from the current OS to the new OS. These procedures are summarized below and differ somewhat from those in the Operator's Manual (Ref 26). For example, the Operator's Manual refers to the program RTGEN instead of RT3GN (RTGEN may have been an earlier version of RT3GN). Also, the Operator's Manual does not discuss the use of an answer file.

```
*RUN,FMGR
:RP,RT3G1
:RP,RT3G2
:RP,RT3G3
:RP,RT3G4
:RP,RT3G5
:RP,RT3G6
:RP,RT3G7
:RP,RT3GN
:EX
*RUN,RT3GN

LIST FILE?
L81914

ECHO?
TR,A81914
  .
  .  (The OS and listing file are generated)
  .
  .
  .
RT3GN FINISHED

*RUN,SWITCH
```

The procedure for using SWTCH is summarized in Appendix B.

Once the OS was executing, the first task was to create a backup disk so that the OS could be recovered in the event of a disk failure. An off-line utility called DSKUP was used to copy the lower platter (where

the OS resides) to the upper platter (which is removable). DSKUP was loaded from the right cassette drive of the graphics terminal (See Appendix F). The procedure for using DSKUP is shown in Appendix B.

One error has been discovered in the most recent answer file. Three memory partitions were specified as 18K, instead of the maximum of 17K, as reported by RT3GN in the list file. This didn't cause any generation errors. These partitions are assumed to be 17K each, so that 3K words of physical memory are unavailable for use.

3.1.2 Capabilities and Limitations

RTE-III provides useful software support for program development and testing. Included are an interactive editor, an assembler, and compilers for BASIC, ALGOL, FORTRAN, and FORTRAN IV. The only compiler that has been used is FORTRAN IV because DSP programs taken from the literature were written in this language. However, two limitations should be noted in light of the DSP application. The primary limitation is the size of logical memory (See Ref 26 for a discussion of physical memory vs. logical memory). DSP programs are typically quite large (the FIR filter design program in Appendix D required 15K words of user partitioned memory), so that many programs cannot be used on this system, unless they are segmented. This is because the maximum partition size for user programs is 17K words. The maximum

partition size is set at generation time, where RT3GN provides the maximum value.

Another notable limitation is that writing an executive program which can reference any of the DSP programs is not as simple as it might be. The simplest way to write an executive program is to write it in FORTRAN IV (or another high order language) and load it into memory with all of its subprograms. This method is not practical for the HP system because logical memory size is relatively limited. Alternate solutions are discussed in chapter five.

3.2 Data Acquisition

As stated previously, data can be acquired in real-time from an A/D converter, or with software that generates data. This section focuses on data obtained which represents real-time phenomena. Examples of software generated data can be found in chapter four. Two related subjects are presented here: The A/D hardware and its software support.

3.2.1 A/D Converter

Two A/D converters were provided for this project. They are the ADC-12QZ model, made by Analog Devices (Ref 6:175-178). Each is mounted on an AC-1548 mounting card. An input range of -10v to +10v, represented by 12 bits in twos-complement format is configured. The bipolar range was chosen to allow more flexibility in

the type of signals that can be analyzed. Twos-complement format was chosen because this is how the HP computer represents numbers in memory. Each A/D converter is connected to an HP 12566B micro-circuit interface card (Ref 16) as shown in Fig. 2.

3.2.2 Software Support for A/D Converter

Four different kinds of software support have been investigated for the A/D converter (Ref 34). The first is polling (i.e., check the I/O channel flag bit) software, whereby the computer initiates an A/D conversion, waits in a loop for the conversion to complete, loads the data into a register, and then stores it in a memory buffer. The second type is direct memory access (DMA). In this case, the computer initializes one channel of the DCPC, which will perform the required operations for data acquisition and storage. The third method is to use a system driver. This method as only partially been implemented. The fourth method involves the use of interrupts. The interrupt method was found to be inappropriate for this application. Each of these methods is discussed with regard to its respective advantages, disadvantages, and implementation. All relevant program listings are in Appendix E.

3.2.2.1 Polling

The main advantage of polling software is that a software delay loop can be included to allow the user

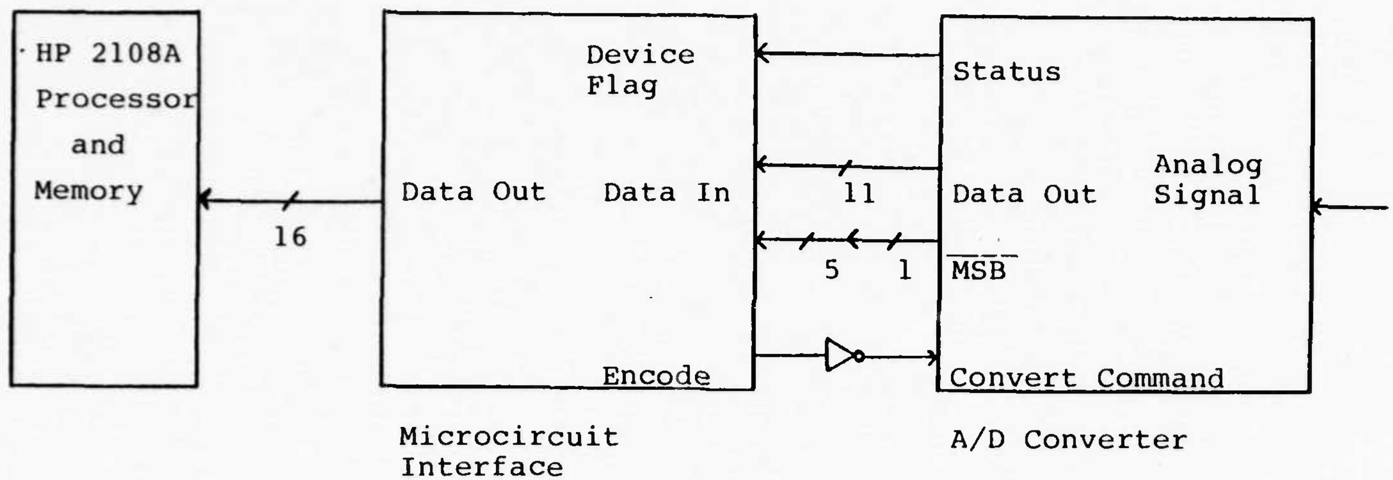


Fig. 2   Connections for A/D Converter and Interface

to interactively specify the sampling rate. In the subroutine WAIT1, the delay is increased by increasing the variable DELAY. Of course, DELAY is just an integer which will cause a different amount of delay for different computers (even of the same type) and different operating conditions (e.g., temperature). In order to relate the delay value to time, a test subroutine (AD14 or AD15) was implemented. This subroutine is nearly identical to the original (AD14B or AD15B), so that the speed of both is approximately the same in the data acquisition loop. The speed of the instructions in this loop will determine the amount of time from the completion of one conversion to the beginning of the next conversion. This time - the instruction execution time (IET) - and the conversion time (CT) can be measured with an oscilloscope by checking the status bit of the A/D converter (Fig. 2, 3). The sum of these two times is the sample time (ST). Several measurements were taken for different values of DELAY, and are summarized in Table I. The measurements are also plotted in Fig. 4. In each case, the CT was about 33 us.

The theoretical sample time (TST) for a periodic signal of a single frequency can be expressed as

$$TST = NC/(f \times NP)$$

where

NC = number of cycles

f = frequency in hertz

NP = number of sampled data points

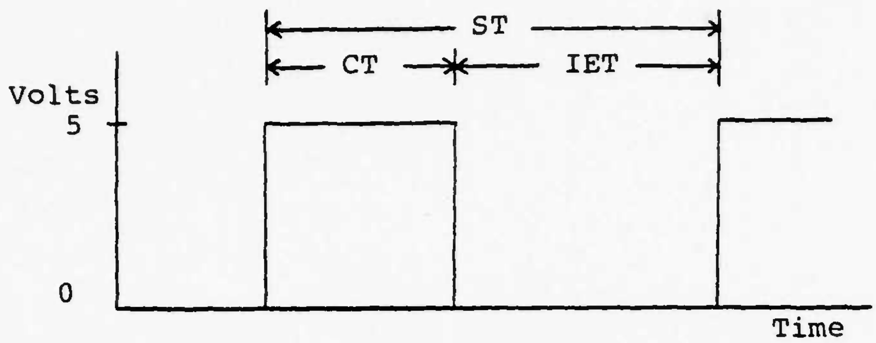


Fig. 3 Oscilloscope Display of Status Bit

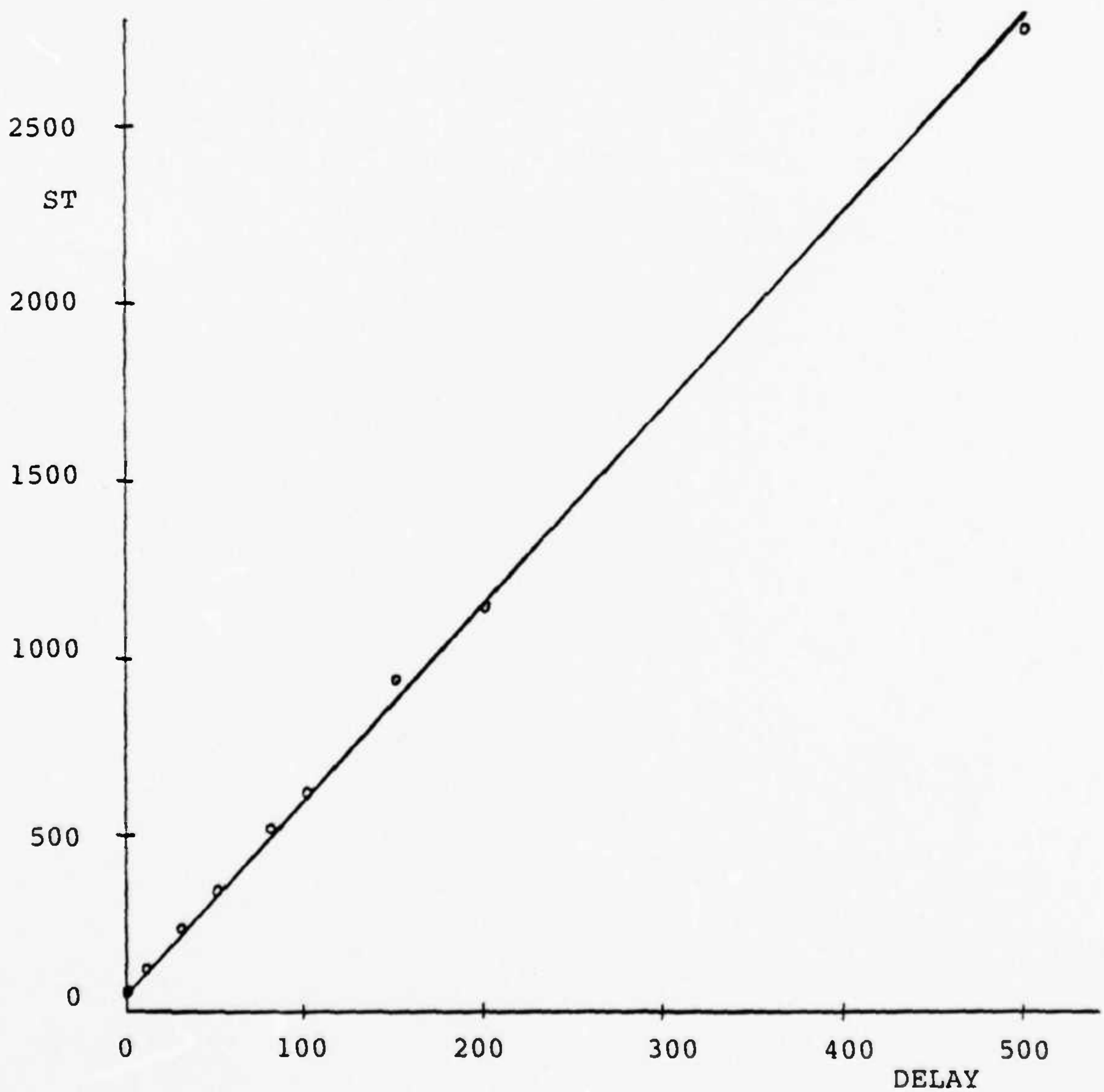


Fig. 4 Plotted Sample Time Measurements

Table I

Sample Time Measurements

DELAY	ST,us
0	76
10	130
30	240
50	350
80	510
100	620
150	890
200	1150
500	2775

Once the TST is calculated, the number for DELAY can be selected from Fig. 4. Data can then be acquired and plotted using this value for DELAY and the value of NP selected above. The actual NC plotted is used to calculated ST. By comparing ST with TST, error in relating DELAY to time can be determined.

The disadvantages of polling software are twofold. First, it is slower than DMA because data must first be transferred to a register. The effect of this is to lower the maximum frequency that can be accurately represented. Secondly, this method precludes real-time processing, whereby one data buffer is being filled while another is being processed (Ref 32). Instead, the processor fills one buffer until it is full.

3.2.2.2 Interrupts

Performing I/O by the use of interrupts is inappropriate for this application for several reasons. First, if a system driver is not used, I/O for the A/D converter can only be accomplished with the interrupt system

turned off. Otherwise, an illegal (not allowed by the OS) interrupt error will occur. Secondly, because of the high speed synchronous nature of the A/D converter, an interrupt must be handled within a certain amount of time. RTE-III has many potential interrupts from other I/O channels that it must handle by priority, so there is no guarantee that data will not be lost from the A/D converter. Thirdly, data which is acquired must first be transferred to a register as with the polling method, so that about the same software overhead is incurred. Therefore, the interrupt method offers no speed advantage over the polling method. If the interrupt method could be used, it would have the advantage that some useful computations could be performed between interrupts.

3.2.2.3 Direct Memory Access

The fastest kind of data acquisition available for this system is direct memory access. The acquisition rate is almost that of the A/D conversion rate. The entire transfer process is handled by hardware (the DCPC) once it has been initialized by the appropriate instructions. Therefore, no software overhead is incurred in acquiring and storing data. Additionally, the processor is free to perform useful computations in parallel with data acquisition. However, the processor speed is reduced slightly because the processor cannot reference memory at the same instant of time that the DCPC is transferring data to or from memory. Thus, the number of computations

during data acquisition can be increased by reducing the number of memory reference instructions. The capability of simultaneous data acquisition and processing permits real-time processing, which is discussed in chapter four.

The only disadvantage of DMA is that the sampling rate cannot be specified interactively by the user, without additional hardware. This hardware might include a programable timer, which is initialized prior to the transfer of each data sample. A count-finished signal would then be substituted for the status signal in Fig. 2.

The subroutine DMA14 (or DMA15) in Appendix E demonstrates the software needed to use DMA. It is called from the main program ANDG2.

3.2.2.4 I/O Driver

A driver is a program designed to support I/O for a particular device. It is placed in the system area at generation time, and gives the OS total control over I/O. Although RTE-III does not provide drivers for the D/A converter card and the microcircuit interface card, user written drivers can be included in the OS generation (Ref 27). Unfortunately, in debugging a driver, each new version requires another generation of the OS. This is a lengthy process even if the generation completes every time with the first attempt.

In order to permit user-written drivers to be implemented and debugged without generating a new OS,

three dummy drivers have been included as part of the latest OS generation. By themselves, these drivers do nothing but return to their calling program. However, by changing the return address in the system area to the beginning address of a user-supplied driver before calling a dummy driver, the actual driver can be tested. A large buffer area is included in the dummy driver so that the actual driver can eventually be copied into this area. This method has not been explored beyond the conceptual stage.

Note that the OS will assign a DCPC channel for the driver. If a DCPC channel is not available, the device goes into a "waiting for DCPC" state (Ref 27:3-16). Because of the possibility of waiting, drivers are probably not suitable for real-time processing.

3.3 Data Storage

The laboratory has two types of devices to meet the requirements of data storage from main memory: the cassette drives in the graphics terminal and the disk drive. The merits and some possible applications of each are presented.

3.3.1 Disk

The disk drive can be used in two different ways to store data. One method is useful for temporary storage while a program is executing. The other method can be used for permanent storage and later retrieval. Both

methods can be useful in DSP applications. Each is demonstrated by programs listed in Appendix G.

Storing data temporarily on a disk involves the Disk Track Allocation EXEC Call (Ref 26:3-18) and the system utility BINRY (Ref 11:3-2). With this utility, the exact area on the disk where the data transfer is to occur must be specified. RTE-III will assign the tracks to the program, and release the tracks when the program terminates. FMGR and other programs cannot access data stored in this manner once the tracks have been released.

One specific application of BINRY is to pass data between two concurrently running programs which are in separate partitions. In addition to the data, some kind of message must also be transferred to signal that the data transfer is complete or in progress. This application has not been implemented.

For permanent storage, FMP calls are used. They allow a user program to create a disk file, and read from or write into a disk file. The data will remain accessible to other programs (such as FMGR) after the user program has terminated. To transfer data among a large system of DSP programs, a standard format for a data file should be established. This would make it easier for a program to find the desired data within any given data file.

3.3.2 Cassette

One way to store program generated data into a disk

file is to first store it on a cassette tape. Although the transfer rate is not as high as that of the disk, data stored on a cassette tape can be accessed by FMGR without the need for a user program to create a disk file. Before the data is sent to the tape (LU 4 or LU 5), tape positioning characters should be sent. After completing the data transfer, an end-of-file (EOF) mark should be written on the tape. These special characters can be sent by using the method described in chapter five.

The data can be transferred to a disk file with a FMGR command as follows:

ST,4,DATA1::20

In this case, the file is transferred from the left cassette to a file called DATA1, which is created by FMGR on cartridge 20 (upper platter disk) (Ref 1:2-23).

3.4 Output Display

Six types of output display can be supported with the system in its current configuration using the following devices: ccnventional CRT terminal, line printer, graphics terminal, graphics printer, oscilloscope, and X-Y plotter (the printers must be connected directly to the graphics terminal). A line printer or graphics printer can be connected to the computer if the RTE-III printer driver is included in a new OS generation.

3.4.1 Conventional CRT Terminal and Line Printer

These types of displays are best suited for listing

tables of values because no special manipulation of the data is required. Graphs and charts can also be implemented, but require additional instructions to draw axes, label axes, scale the data, etc. The CRT has the advantage of speed for checking results quickly. The printer has the advantage of providing a permanent copy of useful results.

3.4.2 Graphics Terminal/Printer

The graphics terminal and graphics printer displays are the best method of implementing graphs and charts because the resolution is only limited by the size of a single point. However, the display size is limited to one screen for the HP 2648A terminal. A graphics printer would simply provide an exact copy of the screen display. The graphics terminal and graphics printer can also serve the purpose of a conventional terminal and printer, respectively. A graphics printer is not currently configured because none is available.

The HP 2648A terminal has a very useful feature for exploiting the graphics capability, called Auto Plot (Ref 15). The only requirements for using it are to fill out the Auto Plot menu, which specifies the data parameters, and then provide the data (usually sent from the computer). These procedures are implemented entirely by the computer in an example in chapter five.

3.4.3 D/A Converter

The D/A converter used for this project is an HP

12555A. It has two analog outputs corresponding to X and Y axes. Each 16-bit word in memory that is transferred is separated into two bytes. Each byte is then converted to an analog signal, ranging from 0 to 10 volts. The D/A converter also has a 20 msec refresh timer for conventional oscilloscopes, and an unblanking signal for storage-type oscilloscopes (Ref 13). The converter is also suitable for driving an X-Y plotter.

3.4.4 Software Support for D/A Converter

The program that has been implemented to support the D/A converter will work for either a conventional oscilloscope or an X-Y plotter (See Appendix I). The inputs to the subroutine DA16B are two real arrays, one containing the X values, and the other containing the Y values. DA16B scales the data into bytes, formats the data into one array, and then outputs it to the D/A converter.

The main differences between the oscilloscope approach and the plotter approach are the refresh timer, which is required for the conventional oscilloscope, and the delay loop, which matches the speed of the computer to that of the plotter. Both devices are currently connected in parallel to the X and Y outputs of the D/A converter.

Chapter 4 Computational Tasks

Four DSP applications programs have been implemented and tested. These programs were selected from the literature (Ref 7) after examining many such programs (Ref 3, 5, 21, 31). The modifications made to allow them to run on the AFIT HP computer were minor and will not be discussed. See Ref 7 for a detailed discussion of the algorithms involved. The four programs that were implemented were chosen for two reasons. First, they are written in ANSI standard FORTRAN IV, so that few changes were expected. The final versions are listed in Appendix D. Secondly, in each case, the author(s) of the program included a test problem. By running the test problem on the HP system and comparing that output to the output provided in the test problem, it was possible to determine if the program was not functioning properly. The outputs generated for all four programs by the HP system were very close (only a few differences in the two least significant digits) to that of the test problem. The differences are assumed to be due to the different accuracies of the computers on which the programs were run. Based on this assumption, the four programs are functioning properly. This chapter concludes with a discussion of a method for implementing real-time processing (RTP). RTP involves simultaneous data acquisition, data processing, and display of results.

4.1 FFT/IFFT

Many variations of the original radix-2 Cooley-Tukey algorithm (Ref 19) have been implemented. This algorithm is commonly called the fast Fourier transform (FFT). The algorithm also computes the inverse of the transform and is called the inverse fast Fourier transform (IFFT). The motivation for using the FFT is that it substantially reduces the required number of machine computations for calculating the DFT. An algorithm used prior to the FFT involved N^2 complex multiplications and approximately N^2 additions (N is the number of sample points). On the other hand, the basic FFT requires at most $(1/2) \times N\log_2 N$ complex multiplications, where N is an integral multiple of two (Ref 5:275).

As an example of the usefulness of the FFT, consider the case where N=8192. Approximately 2.5 minutes of computation time is required to compute the DFT on an IBM 370 using an algorithm which preceeded the FFT. For the same data, the FFT requires approximately 1/4 second (Ref 5:264).

The FFT is an essential capability for a general purpose DSP laboratory. It offers a computationally efficient time-to-frequency transformation. It is also useful for other DSP computations such as convolution, correlation, power spectral density, and digital filter design (Ref 6). The program FAFT4 (See Appendix D) calls a short demonstration version of the FFT (i.e.,

FOURE). FAFT4 generates data for a complex function, computes the theoretical DFT (TDFT), computes the FFT (using FOURE), computes the IFFT (based on values obtained from the TDFT, using FOURE), and outputs these four sets of data. The program has been modified to also list these sets of data in a format suitable for plotting with Auto Plot.

FAFT4 is not intended to be part of the applications software for the laboratory. A more general use of FOURE should be provided which allows the user to interactively specify the data, the number of data points, and whether an FFT or an IFFT is to be computed.

4.2 FIR Filter Design

Digital filtering is an important part of DSP and is in common use. A very important type of filter is the linear phase finite impulse response digital filter (Ref 5:185). One implementation of this is the program FIRF2 (See Appendix D). It uses the Remex exchange algorithm with minimum weighted Chebyshev error in approximating a desired ideal frequency response. The inputs are: 1) filter length (in samples), 2) type of filter (multiple passband/stopband, differentiator, or Hilbert transformer), 3) number of frequency bands, specified by upper and lower cutoff frequencies, 4) desired frequency response in each band, 5) positive weight function in each band, and 6) grid density. The data (provided by the authors) to test the program was

in the following format:

```
55,1,5,0
0.0,0.05,0.1,0.15,0.18,0.25,0.3,0.36,0.41,0.5
0.0,1.0,0.0,1.0,0.0
10.0,1.0,3.0,1.0,20.0
```

Data is read by the program from the left cassette (LU4) of the graphics terminal. For the DSP laboratory the user should be able to interactively specify the data source and enter the data from the keyboard, if that is the source.

4.3 Autocorrelation and Covariance

The program COVAU implements the autocorrelation and covariance methods of linear prediction (LPC) analysis. It consists of three parts: subroutine AUTO (autocorrelation), subroutine COVAR (covariance), and a test program which calls the two subroutines and generates the test input data. AUTO implements a form of Robinson's recursion. COVAR implements a form of Cholesky decomposition. For the DSP laboratory, the user should be able to interactively specify the data source and method of LPC analysis.

4.4 IIR Filter Design

The purpose of program FWIIR is to design finite word-length IIR digital filters. It consists of eight parts: a main program, which acts as an executive, and seven subroutines. Subroutine INDAT reads in all the problem data. FUNCT computes the value of the maximum weighted error function on the coarse grid, and at

the end, on a finer grid. HANDJ is a randomized version of the Hooke and Jeeves Pattern Search. EXPLR is a local exploration program called by HANDJ. SET sets one vector equal to another. SHUFF randomly orders the list of coordinates for EXPLR. UNI generates a random number. For the DSP laboratory, the user should be able to interactively enter the required data, or specify another data source.

4.5 Real-Time Processing

In this context, real-time processing refers to simultaneous data acquisition, data processing, and display of results, without any time limit or interruptions of these three functions. One basic assumption of RTP is that the data can be processed and results displayed at least as fast as it is acquired. Otherwise, some data would be overwritten in memory by new data before the old data was processed. In DSP applications, data is typically processed in time frames (Ref 32). Each time frame is represented by a buffer in main memory. If a time frame is processed as a group of data, then a buffer cannot be processed while it is being filled. With this constraint, at least two buffers are required.

RTP was beyond the scope of this project, but a possible way to implement it on the HP system is shown in a flowchart (See Fig. 5). The program represented by the flowchart is terminated by pressing the "CLEAR DISPLAY" button on the front panel of the computer. This method

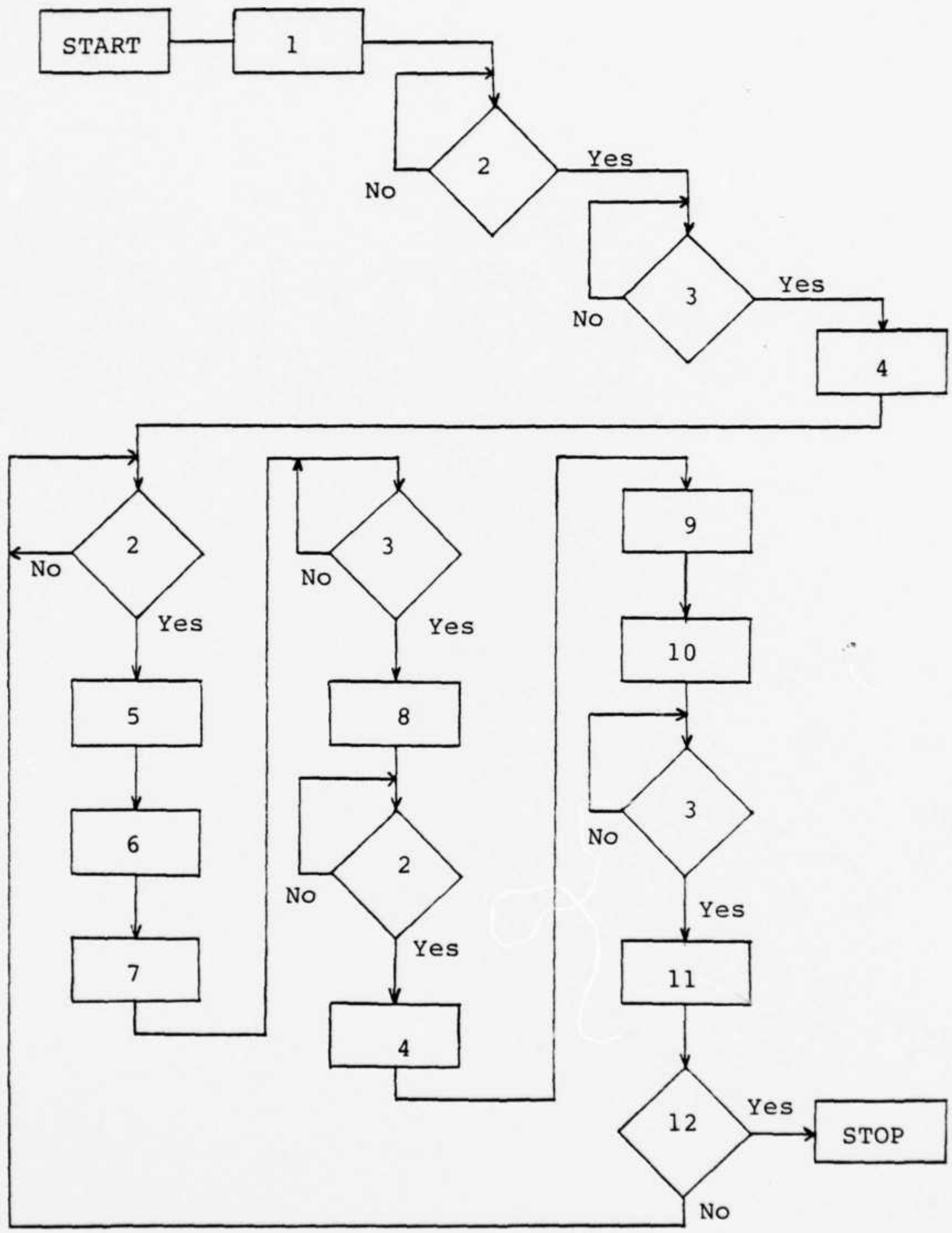


Fig. 5 Flowchart of Real-Time Processing

of program termination has been successfully used with another program (Appendix 1). Each number in the flow-chart corresponds to one of the steps listed below. SR is the switch register displayed on the front panel.

1. Set SR to 1
2. Is DCPC channel 1 ready (flag bit set)?
3. Is DCPC channel 2 ready (flag bit set)?
4. Begin DMA data acquisition in memory buffer #1
5. Begin DMA data acquisition in memory buffer #2
6. Process the data in buffer #1
7. Format the data in buffer #1 for display
8. Begin DMA output from buffer #1
9. Process the data in buffer #2
10. Format the data in buffer #2 for display
11. Begin DMA output from buffer #2
12. Is SR=0?

Some computations may be too lengthy to permit real-time processing. One possible solution to this problem is to replace portions of an assembly language or high-order language program with a microprogram. This involves identifying the "bottle-neck" portions of the program and replacing them with microprogram subroutines which are more time efficient. The system, in its current configuration, supports this capability. See Ref 30 for a complete discussion of the potential for improving program execution speed through microprogramming.

Chapter 5 User Interaction

Several types of software support were implemented to make the DSP laboratory more user-responsive, or to demonstrate its potential in this regard. Chapter three discussed the generation of the OS and the software support provided for the A/D and D/A converters. This chapter first presents a program which demonstrates computer control of the graphics terminal's functions. This is followed by a discussion of a method for executing a program which exceeds the logical memory size of the largest available user partition (17K words). In this way, the user is not constrained to command the execution of several smaller programs to accomplish the same task. Next, two methods are presented whereby one program can schedule another program. A scheduling facility is useful for implementing a DSP executive program which can schedule the DSP applications programs. Finally, a software structure is recommended for a large system of DSP programs.

5.1 Graphics Terminal Control

Since most user interaction will occur with the graphics terminal, an effort was made to relieve the user of many of the manual operations needed to operate the terminal. The program APLOT listed in Appendix H demonstrates a method by which any graphics terminal operation that can be performed manually by the user can also be initiated by the computer. This involves

the use of escape codes (Ref 15). The decimal equivalent of the ASCII character codes are sent from the computer using the Read/Write EXEC call (Ref 26:3-4). APLOT is written in FORTRAN IV, but the characters can also be sent from an assembly language program. The program fills in the Auto Plot menu, draws the axes and grid and plots the program generated data sent from the computer.

5.2 Segmented Program

A segmented program consists of a memory-resident main program and one or more disk-resident subprograms (i.e., segments). By using a segmented structure, the total program size is only limited by the available disk storage space. Data is passed among the segments and the main program through a common storage area within the memory partion allocated for the program. In spite of the size advantage of segmented programs, several limitations should be noted. First, the program will execute slower than if the entire program resided in main memory, because a segment must be loaded from the disk before it can be executed. Therefore, care must be exercised to insure that two or more frequently referenced sections of a program are not chosen to be disk-resident segments. Secondly, the only way to return control to the main program from a segment is for the segment to call the main program (this applies to FORTRAN IV programs, not assembly language). However, this will cause the main program to start executing from the

beginning. Therefore, conditional branching to the segments should occur at or near the beginning of the main program. A third limitation of a segmented program is that if one segment calls another segment, the calling segment is overlayed by the segment it calls.

Writing a segmented program involves the use of the program segment load EXEC call (Ref 26:3-23). An example of this is presented in Appendix J. The following FMGR commands were entered to load the program for execution:

```
:MR,%MAIN1
:MR,%SEGN1
:MR,%SEGM2
:MR,%SEGM3
:RU,LOADR,99,1,,00001
```

To save the loaded program for future execution, the following commands were entered:

```
:SP,MAIN1
:SP,SEGM1
:SP,SEGM2
:SP,SEGM3
```

This saves the program as a type 6 file (ready for execution) on the disk.

5.3 Executive Programs

The two programs discussed in this section are called executive programs because they can schedule other programs, each of which resides in a separate disk file. Both are listed in Appendix K. One method of implementing a DSP executive program (program SCHED) involves the program schedule EXEC call (Ref 26:3-24) and a program suspend EXEC call (Ref 26:3-22). Data is passed through

the system area buffer with a Class I/O - Read/Write EXEC call (Ref 26:3-6) and a Class I/O - Get EXEC call (Ref 26:3-13). Another method of implementing a DSP executive program (program MESG1) involves the Message Processor utility, MESSS (Ref 26:4-40). Any of the system commands can be executed from a FORTRAN IV program by using MESSS (e.g., RU - run command). Data is passed in the same manner described for the previous method.

5.4 Recommended Software Structure

The software structure for the DSP laboratory (demonstrated by program EXECT) should be comprised of a user-oriented executive program and a system of DSP applications programs. The type of executive demonstrated by the program SCHED is considered to be more suitable for the DSP application than that demonstrated by MESG1. SCHED gives the user the option of immediate scheduling so that the output of the scheduled program can be seen before control is returned to the executive. This is not the case for MESG1. EXECT uses the SCHED method. In addition, several of the programs which are scheduled by EXECT are segmented. Therefore, EXECT demonstrates the system's potential for a user-oriented executive program which can schedule any one of a large system of DSP applications programs.

Chapter 6 Recommendations

The system, in its present state, does not qualify as a user-responsive general purpose DSP laboratory, which is the ultimate goal. This is because not all of its current capabilities have been exploited, and because additional capabilities need to be added. Both of these issues will be addressed in this chapter. Implementation of the recommendations which follow should result in a system which meets the design goal as stated above.

6.1 Additional Hardware

The hardware recommended in this section would bring the laboratory much closer to being general purpose. It is intended to support educational and research activities at AFIT. It is also intended to support research for laboratories at WPAFB which might solicit AFIT students for thesis projects on other systems. Therefore, a diversity of equipment is desireable. However, only the addition of a graphics printer is considered essential.

6.1.1 Magnetic Tape Drive

The magnetic tape drive can be useful for at least two reasons. First, information stored on disk can be lost in the event of a disk failure. The information could be later retrieved if it is first backed up on tape. Secondly, a tape can be used to transfer large

amounts of information from one computer system to another (not necessarily HP). For certain types of research, this may be the most efficient means of doing so.

6.1.2 Faster A/D Converter

The A/D converters which are currently configured have a measured conversion time of about 33 us. According to the sampling theorem, a minimum of two samples/cycle of the input signal are required to reconstruct the signal. This corresponds to a maximum frequency of about 15.15 kHz, which does not even cover all of the audio spectrum (20Hz - 20kHz). This limits the types of research related to spectrum analysis that can be conducted on the HP system.

Although HP makes an A/D interface (model 91000A), the sample rate is only 20,000 samples/second. By the above criterion, the maximum useful frequency is 10kHz. Since this is more limited than the existing parts, the HP part is not recommended. A better alternative is the ACD1103 A/D converter, made by Analog Devices. It permits three different conversion times: 1) 8 bits in 1 us, 2) 10 bits in 1.5 us, or 3) 12 bits in 3.5 us (Ref 6). These are maximum values. The comparative maximum useful frequencies are 500 kHz, 333 kHz, and 143 kHz respectively.

Only the conversion time specification has been considered above in recommending an A/D converter.

However, since some applications may emphasize other characteristics such as accuracy and linearity, they should also be considered.

6.1.3 HP Fourier Analyzer

A hardware solution to real-time processing is the HP Fourier Analyzer. It has an oscilloscope display and performs operations such as the FFT/IFFT, convolution, and correlation. Several of the HP users at WPAFB have this device. Conversations with some of the users indicated that it is very useful for real-time DSP.

6.1.4 Graphics Printer

A graphics printer (GP) is an essential addition to the DSP laboratory. It can provide a hard copy for several types of output. For example, displays from the graphics terminal's memory (i.e., graphics memory) can be sent directly to the GP. This output is limited to one screen size. The GP can also serve as a line printer. Any type of output that can be sent to a line printer can also be sent to the GP. From the author's experience, the GP should operate at a serial data transfer rate of at least 300 band in order to qualify as user-responsive.

6.2 DMA for D/A Converter

DMA is the preferred method of sending data to the D/A converter. It can be implemented either as a system driver or in a user program. For RTP, DMA is essential

for a real-time display. No software has yet been written to provide this capability.

6.3 Additional Applications Programs

The four DSP programs which have been implemented (See chapter four) certainly are not sufficient for the diverse computational applications intended for the laboratory. These programs were published by IEEE (Ref 7), and are available on standard 1/2 inch tape from IEEE, along with the other DSP programs in the text. DSP programs can also be found in many other textbooks or periodicals. All of the computational tasks listed in section 2.4 should be implemented.

6.4 Real-Time Processing

Even if an HP Fourier analyzer is configured, it may be desired to perform computations which the analyzer does not provide for. The method for RTP described in chapter five, or some equivalent, should be implemented. RTP is a common activity in DSP and would make the laboratory more general purpose. Therefore, it is an essential capability.

6.5 Operating System

RTE-III has made a significant contribution to the establishment of the HP system as a DSP laboratory. However, because of the limitations previously discussed, RTE-IVB is recommended as an upgrade OS. It allows larger memory partitions (e.g., 28K) for user programs

to be specified (Ref 28:VI-13). It also has a feature called Extended Memory Area (EMA). The EMA is a large area of memory within a partition, limited only by the size of physical memory (Ref 28:VI-14). These are just two of the many advantages of an RTE-IVB OS upgrade. To do this will require the HP 2108A processor board to be modified or replaced. An OS upgrade is not considered essential because RTE-III has been demonstrated to be adequate.

Bibliography

1. Batch-Spool Monitor Reference Manual. Hewlett-Packard Company, May 1979.

2. Bergland, G.D. "A Guided Tour of the Fast Fourier Transform," IEEE Spectrum, 6 (7): 41-52 (July 1969).

3. Cappellini, Vito. Digital Filters and their Applications. New York: Academic Press, 1978.

4. Chen, Chi-Tsong. One-Dimensional Digital Signal Processing. New York: Marcel Dekker, Inc. 1979.

5. Childers, Donald G. and Allen E. Durling. Digital Filtering and Signal Processing. San Francisco: West Publishing Company, 1975.

6. Conversion Products Catalog. Analog Devices, Inc., 1977.

7. Cooley, J.W., et al. Programs for Digital Signal Processing. New York: John Wiley and Sons, Inc., 1979.

8. Cooley, J.W. and J.W. Tukey. "An alogorithm for the Machine Calculation of Complex Fourier Series," Math. Comp., 19 (90): 297-301 (April 1965)

9. Dehning, Waltraud, et al. The Adaptation of Virtual Man-Computer Interfaces to User Requirements in Dialogs. New York: Springer-Verlag, 1981.

10. Despain, Alvin M. "Fourier Transform Computers Using CORDIC Iterations, "IEEE Transactions on Computers," C-23 (10): 993-1001 (October 1974).

11. DOS/RTE Relocatable Library Reference Manual. Hewlett-Packard Company, December 1978.

12. Hodge, Theo. W. and Anita P. Skelton. "A General Purpose Mini-Computer Based Digital Signal Processing Laboratory," NRL Memorandum Report 3502 (May 1977).

13. HP 12555B Digital-to-Analog Converter Interface Kit. Hewlett-Packard Company, March 1973.

14. HP 21MX Computer Series Operator's Manual. Hewlett-Packard Company, July 1974.

15. HP 2648A Graphics Terminal User's Manual. Hewlett-Packard Company, January 1979.

16. HP 12566B Microcircuit Interface Kit. Hewlett-Packard Company, June 1974.

17. Ledgard, Henry, et al. Directions in Human Factors for Interactive Systems. New York: Springer-Verlag, 1981.

18. Liu, Bede, et al. Digital Filters and the Fast Fourier Transform. Stroudsburg: Dowden, Hutchinson & Ross, Inc., 1975.

19. Lorin, H. and H.M. Deitel, Operating Systems. Reading: Addison-Wesley Publishing Company, Inc., 1981.

20. Madnick, Stuart E. and John J. Donovan. Operating Systems. New York: McGraw-Hill Book Company, 1974.

21. Markel, J.D. and A.H. Gray, Jr. Linear Prediction of Speech. New York: Springer-Verlag, 1976.

22. Mehlmann, Marilyn. When People Use Computers. New Jersey: Prentice-Hall, Inc., 1981.

23. Oppenheim, Alan V. Applications of Digital Signal Processing. Englewood Cliffs: Prentice-Hall, Inc., 1978.

24. Papoulis, Athanasios. Signal Analysis. New York: McGraw-Hill, Inc., 1977.

25. Rabiner, L.R. and B. Gold. Theory and Application of Digital Signal Processing. Englewood Cliffs: Prentice-Hall, Inc., 1975.

26. Real-Time Executive III Software System Programming and Operating Manual. Hewlett Packard Company, October 1975.

27. RTE Operating System Driver Writing Manual. Hewlett-Packard Company, January 1980.

28. RTE-IVB Programmer's Reference Manual. Hewlett-Packard Company, January 1980.

29. Smith, H.T., et al. Human Interaction with Computers. New York: Academic Press, 1980.

30. Steidle, John J. "Microprogramming: A Tool to Improve Program Performance," AFIT Thesis, (December 1981)

31. Taylor F. and S. Smith. Digital Signal Processing in FORTRAN. Lexington Books, 1976.

32. _Time Series Systems Operating Manual_. Time/Data Corporation, 1973.

33. Zohar, Stalhav. "Fast Hardware Fourier Transformation Through Counting," _IEEE Transactions on Computers_. _C-22_ (5): 433-441 (May 1973).

34. _21MX M-Series and E-Series Computer I/O Interfacing Guide_. Hewlett-Packard Company, July 1977.

Appendix A: RTE-III Generation Answer File

The generation answer file provides responses to input requests by RT3GN. In this way, the generation of the OS can proceed without interruption for inputs which otherwise would be entered from the system console. Listed in this appendix is the answer file used in the most recent OS generation. Changes for a new OS generation can be made by simply editing the file; the filename is A81914.

```
YES                                  * ECHO
40                                   * #TRACKS IN OUTPUT FILE
2S1914,,10,                          * OUTPUT SYSTEM FILE NAME
7906                                 * DISC MODEL
12                                   * DISC SELECT CODE
300,0,0,2,0,22                       * SUBCHANNEL 0
256,0,2,2,0,6                        * SUBCHANNEL 1
536,132,2,2,0,22                     * SUBCHANNEL 2
/E                                   * TERMINATE SUBCHANNEL DEF
48                                   * #128 WORD SECTGORS/TRK
1                                    * SYSTEM SUBCHANNEL
NO                                   * AUX DISC
11                                   * TMG
0                                    * PRIV INTERRUPT
NO                                   * PRIV DIRVERS ACCESS COMM
YES                                  * FG CORE LOCK
YES                                  * BG CORE LOCK
50                                   * SWAP DELAY
112                                  * MEM SIZE
0                                    * BOOT FILE
MAP MODULES                          * RELOACTE MODULES BY NAME
LINKS IN CURRENT                     * CURRENT PAGE LINKING
*
*   RTE-III SYSTEM MODULES
*
REL,%CR3SY,,10,                 * MEMORY RESIDENT SYSTEM
REL,%RLIB1,,10,                 * RTE/DOS LIBRARY PARTS 1 & 2
REL,%RLIB2,,10,                 * RTE/DOS LIBRARY PART 2
REL,%SYLIB,,10,                 * RTE SYSTEM LIBRARY
REL,%CLIB,,10,                  * RTE COMPILER LIBRARY
REL,%BMLIB,,10,                 * RTE BATCH LIBRARY
REL,%3CMD3,,10,                 * RTE-III COMMAND PROGRAM
REL,%BMPG1,,10,                 * RTE BATCH MONITOR PROGRAM PART 1
REL,%BMPG2,,10,                 * RTE BATCH MONITOR PROGRAM PART 2
REL,%BMPG3,,10,                 * RTE BATCH MONITOR PROGRAM PART 3
REL,%DECAR,,10,                 * RTE DECIMAL STRING ARITHMETIC
REL,%LDR3,,10,                  * RTE III LOADR
REL,%MTM,,10,                   * RTE MULTI-TERMINAL MONITOR
REL,%BPVMP,,10,                 * BPVMP
REL,%FF4.N,,10,                 * FORTRAN IV FORMATER
REL,%DVR00,,10,                 * RTE TTY/PUNCH/PHOTO READER DVR
REL,%DVR36,,10,                 * RTE WCS DRIVER
REL,%3DP43,,10,                 * POWER FAILURE DRIVER
REL,%4DV05,,10,                 * 2648A DRIVER
REL,%DVR32,,10,                 * 7906 DISK DRIVER
REL,%DVR50,,10,                 * DUMMY DRIVER
REL,%DVR51,,10,                 * DUMMY DRIVER
REL,%DVR52,,10,                 * DUMMY DRIVER
REL,%AUTOR,,10,                 * AUTO RESTART
REL,%WHZT3,,10,                 * WHZAT?
REL,%EDITR,,10,                 * EDITOR
REL,%XREF,,10,                  * CROSS-REFERENCE
REL,%ASMB,,10,                  * ASSEMBLER
REL,%SWTCH,,10,                 * SWITCH OPERATING SYSTEMS
```

```
REL,%LU,,10,
REL,%TT,,10,
REL,%PPONG,,10,
REL,%MICRO,,10,                * RTE MICROASSEMBLER
REL,%MXREF,,10,                * RTE MICRO CROSS ASSEMBLER
REL,%WLOAD,,10,                * RTE MICRO LOAD UTILITY ROUTINE
/E
D.RTR,1,1                        * PARAMETERS
$$CMD,3,1
WHZAT,1,5
PPONG,1,32767
ASMB,3
XREF,3
LOADR,3
EDITR,3
AUTOR,3,1
PRMPT,3
R$PN$,3,50
/E                               * TERMINATE
*EAU MACRO'S
.MPY,RP,100200
.DIV,RP,100400
.DLD,RP,104200
.DST,RP,104400
*
* FLOAT MACROS'S
*
.FAD,RP,105000
.FSB,RP,105020
.FMP,RP,105040
.FDV,RP,105060
IFIX,RP,105100
FLOAT,RP,105120
.MVW,RP,105777
Z$DBL,RP,3
/E
25                               * # BLANK ID SEGMENTS
15                               * SHORT ID SEGMENTS
3                                * MAX. # PARTITIONS
100                              * FWA BP LINKAGE
40                               * # OF IO CLASSESS
0                                * # LU MAPPING
35                               * # RN
128,512                          * BUFFER LIMITS(LOW,HIGH)
*
* EQUIPMENT TABLE ENTRIES
*
12,DVR32,D                       * EQT #1
13,DVR05,B,X=13,T=13000          * EQT #2
10,DVR36                         * EQT #3
20,DVR00,B,T=13000               * EQT #4
17,DVR01,B,T=500                 * EQT #5
4,DVR43                          * EQT #6
14,DVR50,D,T=100                 * EQT #7
```

```
15,DVR51,D,T=100                    * EQT #8
16,DVR52,D,T=100                    * EQT #9
/E
*
*
* DEVICE REFERENCE TABLE
*
2,0                                 * LU1 SYS. CONSOLE
1,1                                 * LU2 SYS. DISK(LOWER PLATTER)
0,0                                 * LU3 AUX. DISC
2,1                                 * LU4 L. CTU DRIVE
2,2                                 * LU5 R. CTU DRIVE
0,0                                 * LU6
0,0                                 * LU7
0,0                                 * LU8
0,0                                 * LU9
4,0                                 * LU10 TTY
5,0                                 * LU11 PAPER TAPE READER
0,0                                 * LU12
0,0                                 * LU13
0,0                                 * LU14
3,2                                 * LU15 WCS MODULE 12978A
0,0                                 * LU16
7,0                                 * LU17 MICROCIRCUIT 12566B #1
8,0                                 * LU18 MICROCIRCUIT 12566B #2
9,0                                 * LU19 D/A CONVERTER 12555B
1,0                                 * LU20 DISK SUBCHANNEL 0 (UPP
1,2                                 * LU21 DISC SUBCHANNEL 2 (LOW
0,0                                 * LU22
0,0                                 * LU23
0,0                                 * LU24
6,0                                 * LU25 POWER FAIL
/E
*
* INTERRUPT TABLE
*
4,ENT,$POWR
12,EQT,1
13,EQT,2
17,EQT,5
20,PRG,PRMPT
/E
0
0
YES
YES
20
1,4,BG
2,7,BG
3,13,BG
4,14,BG
5,18,BG
6,18,BG
7,18,BG
```

```
/E
FMGR,13
EDITR,14
LOADR,14
ASMB,15
XREF,15
/E
/E
```

Appendix B: Using SWTCH and DSKUP

SWTCH is a program which is run to switch operating systems. Below are listed the questions that the program will ask and the responses that were entered for the last generation. SP means space; CR means carriage return.

```
FILE NAME OF NEW RTE SYSTEM?
S81914

TARGET CHANNEL FOR NEW SYSTEM
SP CR

TARGET SUBCHANNEL (LOGICAL)/UNIT FOR NEW SYSTEM
SP CR

NOW IS THE TIME TO INSERT CARTRIDGE IN TARGET
SUBCHANNEL UNIT.
SP CR

SAVE FILES AT TARGET?
Y

PURGE TYPE 6 FILES
N

AUTO BOOT-UP?
Y

READY TO TRANSFER.  OK TO PROCEED?
Y
```

DSKUP is an off-line utility which can be loaded from a cassette tape. To load it from the left cassette, set the S Reg. to 041300. Press PRESET, IBL, and RUN. When it is finished loading, set the S Reg. to 13 and the P Reg. to 2. Press PRESET and RUN. Below are listed the questions that the program will ask and the responses that were given to copy the operating system to a removable disk.

```
DISC BACKUP UTILITY
TASK?
CO
SOURCE DISC CHANNEL#?
12
SOURCE DISC TYPE?
7906
SOURCE DISC DRIVER#?
O
TYPE OF CCPY?
FR
RTE OR DOS DISC?
RT
FROM CYLINDER#?
O
# OF TRACKS?
256
# OF SURFACES?
2
STARTING HEAD#?
2
DEST DISC DRIVER#?
O
TO CYLINDER#?
O
# OF SURFACES?
2
STARTING HEAD#?
O
6144 WORD BUFFER DESIRED?
YES
VERIFY?
YES
```

Appendix C: HP users at WPAFB

The people listed below were visited to determine if they had any DSP software currently in use, which might be useful to this project. Another reason for the visits was to observe their hardware configurations with respect to support of DSP applications. Unfortunately, no useful software was found. Those involved with DSP used an HP Fourier Analyzer.

NAME	PHONE	BUILDING	ORGANIZATION
John Bankovskis	56361	622	AFWAL/AARI
Bill Griffin	54016/52789	20	ASD/ENADA
Jim Leonard	53050	23	AFWAL/AAFR/2
L. T. Drzal	52952	32	AFWAL/MLBM
Bob Ballard	52493	24C	AFWAL/FIMN
Irvin F. Luke	52372	18	AFWAL/POOC/1

Appendix D: Computational Tasks Listings

Four source code DSP program listings are contained in this Appendix. They were taken from the literature (Ref 7) and modified to run on the HP system. Also included are the inputs used to test each program (if needed), and the resultant outputs. FAFT4 computes the FFT and IFFT of a complex function. FIRF2 designs linear phase FIR filters. COVAU implements autocorrelation and covariance methods of linear prediction analysis. FWIIR designs finite word-length IIR digital filters. See Ref 7 for a detailed explanation of the algorithms employed.

```
FTN4,L
C
C
      PROGRAM FAFT4
C
C
C-----------------------------------------------------------------
C MAIN PROGRAM:  FOURSUBT
C AUTHOR:        C. M. RADER
C                MIT LINCOLN LABORATORY, LEXINGTON, MA  02173
C
C INPUT:         FUNCTION IS GENERATED BY THE PROGRAM TO TEST
C                SUBROUTINE FOURE
C-----------------------------------------------------------------
C
      COMPLEX W, C(32), D, E, B(32), QB(32), A, F(32)
C
C SEQUENCE LENGTH N=2**MU  USES MU=5  THUS N=32
C METHOD IS TO COMPUTE DFT OF KNOWN FUNCTION  A**I, I=0,1,...,N-1
C OUTPUT OF PROGRAM IS LARGEST DIFFERENCE BETWEEN DFT COMPUTED TWO
C WHEN RUN ON AN IBM 370, THE MAX DIFFERENCE WAS 0.389E-04.
C WHEN RUN ON A HONEYWELL 6080N, THE MAX DIFFERENCE WAS 0.238E-06.
C A MAX DIFFERENCE LARGE COMPARED TO THE ACCURACY OF THE COMPUTER
C WOULD PROBABLY INDICATE A PROGRAM ERROR.
C
      MU = 5
C
C SET UP MACHINE CONSTANT
C
C     IOUTD = I1MACH(2)  <--- THIS STATEMENT IS USELESS TO HP.
C
C SET IOUTD TO THE SYSTEM CONSOLE LU#.
C
      IOUTD = 1
      NN = 2**MU
      TPI =8.*ATAN(1.)
      TPION = TPI/FLOAT(NN)
      W = CMPLX(COS(TPION),-SIN(TPION))
C
C GENERATE A**K AS TEST FUNCTION
C
      A = (.9,.3)
      B(1) = (1.,0.)
      QB(1) = B(1)
      DO 10 K=2,NN
        B(K) = A**(K-1)
        QB(K) = B(K)
10    CONTINUE
C
C PRINT COMPLEX INPUT SEQUENCE
C
```

```
      WRITE (IOUTD,9999)
9999  FORMAT (1X,//////24H COMPLEX INPUT SEQUENCE )
      WRITE (IOUTD,9998) (I,QB(I),I=1,NN)
9998  FORMAT (2(2X, 1H(, I3, 1H), 2E14.6))
C
C B(1) CONTAINS A**0  B(K) CONTAINS A**K-1; ETC
C
C COMPUTE DFT OF B IN CLOSED FORM
C
      D = (1.,0.) - A**NN
      DO 20 K=1,NN
        E = (1.,0.) - A*W**(K-1)
        C(K) = D/E
        F(K) = C(K)
20    CONTINUE
C
C DFT OF B IS (1-A**NN)/(1-AW**K)
C
C NOW COMPUTE DFT OF B USING FOURE
C
      CALL FOURE(B, NN, -1)
C
C PRINT FOURE DFT AND THEORETICAL DFT
C
      WRITE (IOUTD,9997)
9997  FORMAT (1X,//////11H FOURE DFT )
      WRITE (IOUTD,9998) (I,B(I),I=1,NN)
      WRITE (IOUTD,9996)
9996  FORMAT (1X,//////17H THEORETICAL DFT )
      WRITE (IOUTD,9998) (I,C(I),I=1,NN)
C
C FIND MAX DIFFERENCE BETWEEN B AND C
C
      DD = 0.
      DO 30 I=1,NN
        DE = CABS(C(I)-B(I))
        IF (DD.GT.DE) GO TO 30
        DD = DE
30    CONTINUE
C
C COMPUTE INVERSE DFT OF C; (F=C)
C
      CALL FOURE(F, NN, 1)
C
C PRINT INVERSE DFT
C
      WRITE (IOUTD,9995)
9995  FORMAT (1X,//////19H FOURE INVERSE DFT )
      WRITE (IOUTD,9998) (I,F(I),I=1,NN)
C
C COMPUTE MAX. DIFF BETWEEN INPUT AND INVERSE OF THEOR. DFT
C
```

```
      GG = 0.
      DO 40 I=1,NN
        GG1 = CABS(QB(I)-F(I))
        IF (GG1.LE.GG) GO TO 40
        GG =GG1
40    CONTINUE
C
C PRINT MAXIMUM DIFFERENCE
C
      WRITE (IOUTD,9994) DD
      WRITE (IOUTD,9993) GG
9994  FORMAT (/42H MAX DIFF BETWEEN THEOR AND FOURE DFT IS  , E12.3)
9993  FORMAT (/51H MAX DIFF BETWEEN ORIGINAL DATA AND INVERSE DFT IS  ,
     *     E12.3,//////)
      WRITE (4,9000) (QB(I),B(I),C(I),F(I),I=1,NN)
9000  FORMAT(8(2X,F7.5))
      STOP
      END
C
C-----------------------------------------------------------------------
C SUBROUTINE FOURE
C PERFORMS COOLEY-TUKEY FAST FOURIER TRANSFORM
C-----------------------------------------------------------------------
C
      SUBROUTINE FOURE(DATA, N, ISI)
C
C THE COOLEY-TUKEY FAST FOURIER TRANSFORM IN ANSI FORTRAN
C
C DATA IS A ONE-DIMENSIONAL COMPLEX ARRAY WHOSE LENGTH, N, IS A
C POWER OF TWO.  ISI IS +1 FOR AN INVERSE TRANSFORM AND -1 FOR A
C FORWARD TRANSFORM.  TRANSFORM VALUES ARE RETURNED IN THE INPUT
C ARRAY, REPLACING THE INPUT.
C TRANSFORM(J)=SUM(DATA(I)*W**((I-1)*(J-1))), WHERE I AND J RUN
C FROM 1 TO N AND W = EXP (ISI*2*PI*SQRT(-1)/N).  PROGRAM ALSO
C COMPUTES INVERSE TRANSFORM, FOR WHICH THE DEFINING EXPRESSION
C IS INVTR(J)=(1/N*SUM(DATA(I)*W**((I-1)*(J-1))).
C RUNNING TIME IS PROPORTIONAL TO N*LOG2(N), RATHER THAN TO THE
C CLASSICAL N**2.
C AFTER PROGRAM BY BRENNER, JUNE 1967.  THIS IS A VERY SHORT VERSION
C OF THE FFT AND IS INTENDED MAINLY FOR DEMONSTRATION.  PROGRAMS
C ARE AVAILABLE IN THIS COLLECTION WHICH RUN FASTER AND ARE NOT
C RESTRICTED TO POWERS OF 2 OR TO ONE-DIMENSIONAL ARRAYS.
C SEE -- IEEE TRANS AUDIO (JUNE 1967), SPECIAL ISSUE ON FFT.
C
      COMPLEX DATA(N)
      COMPLEX TEMP, W
C     IOUTD = I1MACH(2)   <--- THIS STATEMENT USELESS TO HP.
C
C SET IOUTD TO SYSTEM CONSOLE LU#.
C
      IOUTD = 1
```

```
C
C CHECK FOR POWER OF TWO UP TO 15
C
      NN = 1
      DO 10 I=1,15
        M= I
        NN = NN*2
        IF (NN.EQ.N) GO TO 20
10    CONTINUE
      WRITE (IOUTD,9999)
9999  FORMAT (30H N NOT A POWER OF 2 FOR FOURE )
      STOP
20    CONTINUE
C
      PI = 4.*ATAN(1.)
      FN = FLOAT(N)
C
C THIS SECTION PUTS DATA IN BIT-REVERSED ORDER
C
      J = 1
      DO 80 I=1,N
C
C AT THIS POINT, I AND J ARE A BIT REVERSED PAIR (EXCEPT FOR THE
C DISPLACEMENT OF +1)
C
      IF (I-J) 30, 40, 40
C
C EXCHANGE DATA(I) WITH DATA(J) IF I.LT.J.
C
30    TEMP = DATA(J)
      DATA(J) = DATA(I)
      DATA(I) = TEMP
C
C IMPLEMENT J=J+1, BIT-REVERSED COUNTER
C
40    M = N/2
50    IF (J-M) 70, 70, 60
60    J = J - M
      M = (M+1)/2
      GO TO 50
70    J = J + M
80    CONTINUE
C
C NOW COMPUTE THE BUTTERFLIES
C
      MMAX = 1
90    IF (MMAX-N) 100, 130, 130
100   ISTEP = 2*MMAX
      DO 120 M=1,MMAX
        THETA = PI*FLOAT(ISI*(M-1))/FLOAT(MMAX)
        W = CMPLX(COS(THETA),SIN(THETA))
```

```
          DO 110 I=M,N,ISTEP
            J = I + MMAX
            TEMP = W*DATA(J)
            DATA(J) = DATA(I) - TEMP
            DATA(I) = DATA(I) + TEMP
110       CONTINUE
120     CONTINUE
        MMAX = ISTEP
        GO TO 90
130     IF (ISI) 160, 140, 140
C
C FOR INV TRANS -- ISI=1 -- MULTIPLY OUTPUT BY 1/N
C
140     DO 150 I=1,N
          DATA(I) = DATA(I)/FN
150     CONTINUE
160     RETURN
        END
```

FAFT4 Original Output

COMPLEX INPUT SEQUENCE

(1)	.100000E+01	.000000E+00	(2)	.900000E+00	.300000E+00
(3)	.720000E+00	.540000E+00	(4)	.486000E+00	.702000E+00
(5)	.226800E+00	.777600E+00	(6)	-.291600E-01	.767880E+00
(7)	-.256608E+00	.682344E+00	(8)	-.435650E+00	.537127E+00
(9)	-.553223E+00	.352719E+00	(10)	-.603717E+00	.151480E+00
(11)	-.588789E+00	-.447828E-01	(12)	-.516475E+00	-.216941E+00
(13)	-.399745E+00	-.350190E+00	(14)	-.254714E+00	-.435094E+00
(15)	-.987142E-01	-.467999E+00	(16)	.515569E-01	-.450813E+00
(17)	.181645E+00	-.390265E+00	(18)	.280560E+00	-.296745E+00
(19)	.341528E+00	-.182902E+00	(20)	.362245E+00	-.621538E-01
(21)	.344667E+00	.527352E-01	(22)	.294380E+00	.150862E+00
(23)	.219683E+00	.224090E+00	(24)	.130488E+00	.267586E+00
(25)	.371635E-01	.279973E+00	(26)	-.505448E-01	.263125E+00
(27)	-.124428E+00	.221649E+00	(28)	-.178480E+00	.162156E+00
(29)	-.209279E+00	.923963E-01	(30)	-.216070E+00	.203731E-01
(31)	-.200575E+00	-.464851E-01	(32)	-.166572E+00	-.102009E+00

FOURE DFT

(1)	.693974E+00	.349971E+01	(2)	.279227E+01	.805045E+01
(3)	.940297E+01	-.913500E+01	(4)	.186645E+01	-.383383E+01
(5)	.113182E+01	-.223416E+01	(6)	.904795E+00	-.153463E+01
(7)	.799557E+00	-.113961E+01	(8)	.739606E+00	-.882314E+00
(9)	.700862E+00	-.698565E+00	(10)	.673575E+00	-.558478E+00
(11)	.653109E+00	-.446246E+00	(12)	.636991E+00	-.352689E+00
(13)	.623788E+00	-.272086E+00	(14)	.612613E+00	-.200642E+00
(15)	.602883E+00	-.135703E+00	(16)	.594201E+00	-.753133E-01
(17).	.586277E+00	-.179496E-01	(18)	.578900E+00	.376520E-01
(19)	.571899E+00	.926065E-01	(20)	.565136E+00	.147983E+00
(21)	.558492E+00	.204881E+00	(22)	.551859E+00	.264522E+00
(23)	.545133E+00	.328364E+00	(24)	.538214E+00	.398257E+00
(25)	.531001E+00	.476678E+00	(26)	.523403E+00	.567132E+00
(27)	.515362E+00	.674849E+00	(28)	.506926E+00	.808101E+00
(29)	.498467E+00	.980906E+00	(30)	.491389E+00	.121921E+01
(31)	.490733E+00	.157708E+01	(32)	.517354E+00	.218883E+01

THEORETICAL DFT

(1)	.693974E+00	.349971E+01	(2)	.279227E+01	.805045E+01
(3)	.940298E+01	-.913501E+01	(4)	.186645E+01	-.383383E+01
(5)	.113182E+01	-.223416E+01	(6)	.904794E+00	-.153463E+01
(7)	.799557E+00	-.113961E+01	(8)	.739606E+00	-.882315E+00
(9)	.700862E+00	-.698566E+00	(10)	.673576E+00	-.558479E+00
(11)	.653110E+00	-.446245E+00	(12)	.636991E+00	-.352689E+00
(13)	.623788E+00	-.272086E+00	(14)	.612613E+00	-.200642E+00
(15)	.602883E+00	-.135703E+00	(16)	.594200E+00	-.753140E-01
(17)	.586277E+00	-.179496E-01	(18)	.578900E+00	.376514E-01
(19)	.571899E+00	.926066E-01	(20)	.565136E+00	.147983E+00
(21)	.558492E+00	.204880E+00	(22)	.551859E+00	.264522E+00
(23)	.545134E+00	.328364E+00	(24)	.538215E+00	.398257E+00
(25)	.531002E+00	.476677E+00	(26)	.523404E+00	.567132E+00
(27)	.515363E+00	.674849E+00	(28)	.506927E+00	.808100E+00
(29)	.498468E+00	.980905E+00	(30)	.491391E+00	.121920E+01
(31)	.490734E+00	.157708E+01	(32)	.517358E+00	.218883E+01

FOURE INVERSE DFT

(1)	.100000E+01	-.774860E-06	(2)	.900000E+00	.299999E+00
(3)	.720000E+00	.540000E+00	(4)	.486000E+00	.702000E+00
(5)	.226800E+00	.777600E+00	(6)	-.291604E-01	.767880E+00
(7)	-.256608E+00	.682344E+00	(8)	-.435651E+00	.537127E+00
(9)	-.553224E+00	.352719E+00	(10)	-.603717E+00	.151480E+00
(11)	-.588790E+00	-.447831E-01	(12)	-.516476E+00	-.216942E+00
(13)	-.399746E+00	-.350190E+00	(14)	-.254714E+00	-.435095E+00
(15)	-.987140E-01	-.467999E+00	(16)	.515572E-01	-.450814E+00
(17)	.181646E+00	-.390265E+00	(18)	.280561E+00	-.296745E+00
(19)	.341528E+00	-.182902E+00	(20)	.362246E+00	-.621535E-01
(21)	.344667E+00	.527359E-01	(22)	.294380E+00	.150862E+00
(23)	.219683E+00	.224090E+00	(24)	.130488E+00	.267586E+00
(25)	.371634E-01	.279974E+00	(26)	-.505450E-01	.263126E+00
(27)	-.124428E+00	.221649E+00	(28)	-.178480E+00	.162156E+00
(29)	-.209278E+00	.923962E-01	(30)	-.216069E+00	.203729E-01
(31)	-.200574E+00	-.464854E-01	(32)	-.166571E+00	-.102009E+00

MAX DIFF BETWEEN THEOR AND FOURE DFT IS , .111E-04

MAX DIFF BETWEEN ORIGINAL DATA AND INVERSE DFT IS .105E-05

FAFT4 Output in Auto Plot Format

1.00000	.00000	.69397	3.49971	.69397	3.49971	1.00000	-.00000
.90000	.30000	2.79227	8.05045	2.79227	0.05045	.90000	.30000
.72000	.54000	9.40297	-9.1350	9.40298	-9.1350	.72000	.54000
.48600	.70200	1.86645	-3.8338	1.86645	-3.8338	.48600	.70200
.22680	.77760	1.13182	-2.2342	1.13182	-2.2342	.22680	.77760
-.02916	.76788	.90479	-1.5346	.90479	-1.5346	-.02916	.76788
-.25661	.68234	.79956	-1.1396	.79956	-1.1396	-.25661	.68234
-.43565	.53713	.73961	-.88231	.73961	-.88231	-.43565	.53713
-.55322	.35272	.70086	-.69857	.70086	-.69857	-.55322	.35272
-.60372	.15148	.67358	-.55848	.67358	-.55848	-.60372	.15148
-.58879	-.04478	.65311	-.44625	.65311	-.44625	-.58879	-.04478
-.51648	-.21694	.63699	-.35269	.63699	-.35269	-.51648	-.21694
-.39975	-.35019	.62379	-.27209	.62379	-.27209	-.39975	-.35019
-.25471	-.43509	.61261	-.20064	.61261	-.20064	-.25471	-.43509
-.09871	-.46000	.60288	-.13570	.60288	-.13570	-.09871	-.46000
.05156	-.45081	.59420	-.07531	.59420	-.07531	.05156	-.45081
.18165	-.39026	.58628	-.01795	.58628	-.01795	.18165	-.39026
.28056	-.29674	.57890	.03765	.57890	.03765	.28056	-.29674
.34153	-.18290	.57190	.09261	.57190	.09261	.34153	-.18290
.36225	-.06215	.56514	.14798	.56514	.14798	.36225	-.06215
.34467	.05274	.55849	.20488	.55849	.20488	.34467	.05274
.29438	.15086	.55186	.26452	.55186	.26452	.29438	.15086
.21968	.22409	.54513	.32836	.54513	.32836	.21968	.22409
.13049	.26759	.53821	.39826	.53821	.39826	.13049	.26759
.03716	.27997	.53100	.47660	.53100	.47668	.03716	.27997
-.05054	.26313	.52340	.56713	.52340	.56713	-.05054	.26313
-.12443	.22165	.51536	.67485	.51536	.67485	-.12443	.22165
-.17848	.16216	.50693	.80810	.50693	.80810	-.17848	.16216
-.20928	.09240	.49847	.98091	.49847	.98090	-.20928	.09240
-.21607	.02037	.49139	1.21921	.49139	1.21920	-.21607	.02037
-.20057	-.04649	.49073	1.57708	.49073	1.57708	-.20057	-.04649
-.16657	-.10201	.51735	2.18083	.51736	2.18083	-.16657	-.10201

```
FTN4,L
C
C
      PROGRAM FIRF2
C
C
C-----------------------------------------------------------------------
C MAIN PROGRAM: FIR LINEAR PHASE FILTER DESIGN PROGRAM
C
C AUTHORS: JAMES H. MCCLELLAN
C          DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIE
C          MASSACHUSETTS INSTITUTE OF TECHNOLOGY
C          CAMBRIDGE, MASS. 02139
C
C          THOMAS W. PARKS
C          DEPARTMENT OF ELECTRICAL ENGINEERING
C          RICE UNIVERSITY
C          HOUSTON, TEXAS  77001
C
C          LAWRENCE R. RABINER
C          BELL LABORATORIES
C          MURRAY HILL, NEW JERSEY  07974
C
C INPUT:
C  NFILT-- FILTER LENGTH
C  JTYPE-- TYPE OF FILTER
C          1 = MULTIPLE PASSBAND/STOPBAND FILTER
C          2 = DIFFERENTIATOR
C          3 = HILBERT TRANSFORM FILTER
C  NBANDS-- NUMBER OF BANDS
C  LGRID-- GRID DENSITY, WILL BE SET TO 16 UNLESS
C          SPECIFIED OTHERWISE BY A POSITIVE CONSTANT.
C
C  EDGE(2*NBANDS)-- BANDEDGE ARRAY, LOWER AND UPPER EDGES FOR EACH
C                   WITH A MAXIMUM OF 10 BANDS.
C
C  FX(NBANDS)-- DESIRED FUNCTION ARRAY (OR DESIRED SLOPE IF A
C               DIFFERENTIATOR) FOR EACH BAND.
C
C  WTX(NBANDS)-- WEIGHT FUNCTION ARRAY IN EACH BAND.  FOR A
C                DIFFERENTIATOR, THE WEIGHT FUNCTION IS INVERSELY
C                PROPORTIONAL TO F.
C
C  SAMPLE INPUT DATA SETUP:
C       32,1,3,0
C       0.0,0.1,0.2,0.35,0.425,0.5
C       0.0,1.0,0.0
C       10.0,1.0,10.0
C     THIS DATA SPECIFIES A LENGTH 32 BANDPASS FILTER WITH
C     STOPBANDS 0 TO 0.1 AND 0.425 TO 0.5, AND PASSBAND FROM
C     0.2 TO 0.35 WITH WEIGHTING OF 10 IN THE STOPBANDS AND 1
```

```
C      IN THE PASSBAND.  THE GRID DENSITY DEFAULTS TO 16.
C      THIS IS THE FILTER IN FIGURE 10.
C
C      THE FOLLOWING INPUT DATA SPECIFIES A LENGTH 32 FULLBAND
C      DIFFERENTIATOR WITH SLOPE 1 AND WEIGHTING OF 1/F.
C      THE GRID DENSITY WILL BE SET TO 20.
C        32,2,1,20
C        0,0.5
C        1.0
C        1.0
C
C-----------------------------------------------------------------
C
       COMMON PI2,AD,DEV,X,Y,GRID,DES,WT,ALPHA,IEXT,NFCNS,NGRID
       COMMON NITER,IOUT
       DIMENSION IEXT(34),AD(34),ALPHA(34),X(34),Y(34)
       DIMENSION H(34)
       DIMENSION DES(544),GRID(544),WT(544)
       DIMENSION EDGE(20),FX(10),WTX(10),DEVIAT(10)
       DOUBLE PRECISION PI2,PI
       DOUBLE PRECISION AD,DEV,X,Y
       DOUBLE PRECISION GEE,D
       INTEGER BD1,BD2,BD3,BD4
       DATA BD1,BD2,BD3,BD4/1HB,1HA,1HN,1HD/
C      INPUT=I1MACH(1)
C      IOUT=I1MACH(2)
       INPUT=4
       IOUT=1
       WRITE(IOUT,9000)
9000   FORMAT(22(/))
       PI=4.0*DATAN(1.0D0)
       PI2=2.0D00*PI
C
C THE PROGRAM IS SET UP FOR A MAXIMUM LENGTH OF 64, BUT
C THIS UPPER LIMIT CAN BE CHANGED BY REDIMENSIONING THE
C ARRAYS IEXT, AD, ALPHA, X, Y, H TO BE NFMAX/2 + 2.
C THE ARRAYS DES, GRID, AND WT MUST DIMENSIONED
C 16(NFMAX/2 + 2).
C
       NFMAX=64
100    CONTINUE
       JTYPE=0
C
C PROGRAM INPUT SECTION
C
       READ(INPUT,*) NFILT,JTYPE,NBANDS,LGRID
       IF(NFILT.EQ.0) STOP
110    FORMAT(4I5)
       IF(NFILT.LE.NFMAX.OR.NFILT.GE.3) GO TO 115
       CALL ERROR
       STOP
```

```
115   IF(NBANDS.LE.0) NBANDS=1
C
C GRID DENSITY IS ASSUMED TO BE 16 UNLESS SPECIFIED
C OTHERWISE
C
      IF(LGRID.LE.0) LGRID=16
      JB=2*NBANDS
      READ(INPUT,*) (EDGE(J),J=1,JB)
120   FORMAT(4F15.9)
      READ(INPUT,*) (FX(J),J=1,NBANDS)
      READ(INPUT,*) (WTX(J),J=1,NBANDS)
      IF(JTYPE.GT.0.AND.JTYPE.LE.3) GO TO 125
      CALL ERROR
      STOP
125   NEG=1
      IF(JTYPE.EQ.1) NEG=0
      NODD=NFILT/2
      NODD=NFILT-2*NODD
      NFCNS=NFILT/2
      IF(NODD.EQ.1.AND.NEG.EQ.0) NFCNS=NFCNS+1
C
C SET UP THE DENSE GRID.  THE NUMBER OF POINTS IN THE GRID
C IS (FILTER LENGTH + 1)*GRID DENSITY/2
C
      GRID(1)=EDGE(1)
      DELF=LGRID*NFCNS
      DELF=0.5/DELF
      IF(NEG.EQ.0) GO TO 135
      IF(EDGE(1).LT.DELF) GRID(1)=DELF
135   CONTINUE
      J=1
      L=1
      LBAND=1
140   FUP=EDGE(L+1)
145   TEMP=GRID(J)
C
C CALCULATE THE DESIRED MAGNITUDE RESPONSE AND THE WEIGHT
C FUNCTION ON THE GRID
C
      DES(J)=EFF(TEMP,FX,WTX,LBAND,JTYPE)
      WT(J)=WATE(TEMP,FX,WTX,LBAND,JTYPE)
      J=J+1
      GRID(J)=TEMP+DELF
      IF(GRID(J).GT.FUP) GO TO 150
      GO TO 145
150   GRID(J-1)=FUP
      DES(J-1)=EFF(FUP,FX,WTX,LBAND,JTYPE)
      WT(J-1)=WATE(FUP,FX,WTX,LBAND,JTYPE)
      LBAND=LBAND+1
      L=L+2
      IF(LBAND.GT.NBANDS) GO TO 160
```

```
      GRID(J)=EDGE(L)
      GO TO 140
160   NGRID=J-1
      IF(NEG.NE.NODD) GO TO 165
      IF(GRID(NGRID).GT.(0.5-DELF)) NGRID=NGRID-1
165   CONTINUE
C
C SET UP A NEW APPROXIMATION PROBLEM WHICH IS EQUIVALENT
C TO THE ORIGINAL PROBLEM
C
      IF(NEG) 170,170,180
170   IF(NODD.EQ.1) GO TO 200
      DO 175 J=1,NGRID
      CHANGE=DCOS(PI*GRID(J))
      DES(J)=DES(J)/CHANGE
175   WT(J)=WT(J)*CHANGE
      GO TO 200
180   IF(NODD.EQ.1) GO TO 190
      DO 185 J=1,NGRID
      CHANGE=DSIN(PI*GRID(J))
      DES(J)=DES(J)/CHANGE
185   WT(J)=WT(J)*CHANGE
      GO TO 200
190   DO 195 J=1,NGRID
      CHANGE=DSIN(PI2*GRID(J))
      DES(J)=DES(J)/CHANGE
195   WT(J)=WT(J)*CHANGE
C
C INITIAL GUESS FOR THE EXTREMAL FREQUENCIES--EQUALLY
C SPACED ALONG THE GRID
C
200   TEMP=FLOAT(NGRID-1)/FLOAT(NFCNS)
      DO 210 J=1,NFCNS
      XT=J-1
210   IEXT(J)=XT*TEMP+1.0
      IEXT(NFCNS+1)=NGRID
      NM1=NFCNS-1
      NZ=NFCNS+1
C
C CALL THE REMEZ EXCHANGE ALGORITHM TO DO THE APPROXIMATION
C PROBLEM
C
      CALL REMEZ
C
C CALCULATE THE IMPULSE RESPONSE.
C
      IF(NEG) 300,300,320
300   IF(NODD.EQ.0) GO TO 310
      DO 305 J=1,NM1
      NZMJ=NZ-J
305   H(J)=0.5*ALPHA(NZMJ)
```

```
      H(NFCNS)=ALPHA(1)
      GO TO 350
310   H(1)=0.25*ALPHA(NFCNS)
      DO 315 J=2,NM1
      NZMJ=NZ-J
      NF2J=NFCNS+2-J
315   H(J)=0.25*(ALPHA(NZMJ)+ALPHA(NF2J))
      H(NFCNS)=0.5*ALPHA(1)+0.25*ALPHA(2)
      GO TO 350
320   IF(NODD.EQ.0) GO TO 330
      H(1)=0.25*ALPHA(NFCNS)
      H(2)=0.25*ALPHA(NM1)
      DO 325 J=3,NM1
      NZMJ=NZ-J
      NF3J=NFCNS+3-J
325   H(J)=0.25*(ALPHA(NZMJ)-ALPHA(NF3J))
      H(NFCNS)=0.5*ALPHA(1)-0.25*ALPHA(3)
      H(NZ)=0.0
      GO TO 350
330   H(1)=0.25*ALPHA(NFCNS)
      DO 335 J=2,NM1
      NZMJ=NZ-J
      NF2J=NFCNS+2-J
335   H(J)=0.25*(ALPHA(NZMJ)-ALPHA(NF2J))
      H(NFCNS)=0.5*ALPHA(1)-0.25*ALPHA(2)
C
C PROGRAM OUTPUT SECTION.
C
350   WRITE(IOUT,360)
360   FORMAT(10(/),70(1H*)//15X,29FINITE IMPULSE RESPONSE (FIR)/
     113X,34HLINEAR PHASE DIGITAL FILTER DESIGN/
     217X,24HREMEZ EXCHANGE ALGORITHM/)
      IF(JTYPE.EQ.1) WRITE(IOUT,365)
365   FORMAT(22X,15HBANDPASS FILTER/)
      IF(JTYPE.EQ.2) WRITE(IOUT,370)
370   FORMAT(22X,14HDIFFERENTIATOR/)
      IF(JTYPE.EQ.3) WRITE(IOUT,375)
375   FORMAT(20X,19HHILBERT TRANSFORMER/)
      WRITE(IOUT,378) NFILT
378   FORMAT(20X,16HFILTER LENGTH = ,I3/)
      WRITE(IOUT,380)
380   FORMAT(15X,28H***** IMPULSE RESPONSE *****)
      DO 381 J=1,NFCNS
      K=NFILT+1-J
      IF(NEG.EQ.0) WRITE(IOUT,382) J,H(J),K
      IF(NEG.EQ.1) WRITE(IOUT,383) J,H(J),K
381   CONTINUE
382   FORMAT(13X,2HH(,I2,4H) = ,E15.8,5H = H(,I3,1H))
383   FORMAT(13X,2HH(,I2,4H) = ,E15.8,6H = -H(,I3,1H))
      IF(NEG.EQ.1.AND.NODD.EQ.1) WRITE(IOUT,384) NZ
384   FORMAT(13X,2HH(,I2,8H) = 0.0)
```

```
      DO 450 K=1,NBANDS,4
      KUP=K+3
      IF(KUP.GT.NBANDS) KUP=NBANDS
      WRITE(IOUT,385) (BD1,BD2,BD3,BD4,J,J=K,KUP)
385   FORMAT(/24X,4(4A1,I3,7X))
      WRITE(IOUT,390) (EDGE(2*J-1),J=K,KUP)
390   FORMAT(2X,15HLOWER BAND EDGE,5F14.7)
      WRITE(IOUT,395) (EDGE(2*J),J=K,KUP)
395   FORMAT(2X,15HUPPER BAND EDGE,5F14.7)
      IF(JTYPE.NE.2) WRITE(IOUT,400) (FX(J),J=K,KUP)
400   FORMAT(2X,13HDESIRED VALUE,2X,5F14.7)
      IF(JTYPE.EQ.2) WRITE(IOUT,405) (FX(J),J=K,KUP)
405   FORMAT(2X,13HDESIRED SLOPE,2X,5F14.7)
      WRITE(IOUT,410) (WTX(J),J=K,KUP)
410   FORMAT(2X,9HWEIGHTING,6X,5F14.7)
      DO 420 J=K,KUP
420   DEVIAT(J)=DEV/WTX(J)
      WRITE(IOUT,425) (DEVIAT(J),J=K,KUP)
425   FORMAT(2X,9HDEVIATION,6X,5F14.7)
      IF(JTYPE.NE.1) GO TO 450
      DO 430 J=K,KUP
430   DEVIAT(J)=20.0*ALOGT(DEVIAT(J)+FX(J))
      WRITE(IOUT,435) (DEVIAT(J),J=K,KUP)
435   FORMAT(2X,15HDEVIATION IN DB,5F14.7)
450   CONTINUE
      DO 452 J=1,NZ
      IX=IEXT(J)
452   GRID(J)=GRID(IX)
      WRITE(IOUT,455) (GRID(J),J=1,NZ)
455   FORMAT(/2X,47HEXTREMAL FREQUENCIES--MAXIMA OF THE ERROR CURVE/
     1 (2X,5F12.7))
      WRITE(IOUT,460)
460   FORMAT(/1X,70(1H*),10(/))
C     GO TO 100
      STOP
      END
C
C-----------------------------------------------------------------------
C FUNCTION: EFF
C   FUNCTION TO CALCULATE THE DESIRED MAGNITUDE RESPONSE
C   AS A FUNCTION OF FREQUENCY.
C   AN ARBITRARY FUNCTION OF FREQUENCY CAN BE
C   APPROXIMATED IF THE USER REPLACES THIS FUNCTION
C   WITH THE APPROPRIATE CODE TO EVALUATE THE IDEAL
C   MAGNITUDE.  NOTE THAT THE PARAMETER FREQ IS THE
C   VALUE OF NORMALIZED FREQUENCY NEEDED FOR EVALUATION
C-----------------------------------------------------------------------
C
      FUNCTION EFF(FREQ,FX,WTX,LBAND,JTYPE)
      DIMENSION FX(5),WTX(5)
      IF(JTYPE.EQ.2) GO TO 1
```

```
      EFF=FX(LBAND)
      RETURN
1     EFF=FX(LBAND)*FREQ
      RETURN
      END
C
C-----------------------------------------------------------------------
C FUNCTION: WATE
C   FUNCTION TO CALCULATE THE WEIGHT FUNCTION AS A FUNCTION
C   OF FREQUENCY.  SIMILAR TO THE FUNCTION EFF, THIS FUNCTION CAN
C   BE REPLACED BY A USER-WRITTEN ROUTINE TO CALCULATE ANY
C   DESIRED WEIGHTING FUNCTION.
C-----------------------------------------------------------------------
C
      FUNCTION WATE(FREQ,FX,WTX,LBAND,JTYPE)
      DIMENSION FX(5),WTX(5)
      IF(JTYPE.EQ.2) GO TO 1
      WATE=WTX(LBAND)
      RETURN
1     IF(FX(LBAND).LT.0.0001) GO TO 2
      WATE=WTX(LBAND)/FREQ
      RETURN
2     WATE=WTX(LBAND)
      RETURN
      END
C
C-----------------------------------------------------------------------
C SUBROUTINE ERROR
C   THIS ROUTINE WRITES AN ERROR MESSAGE IF AN
C   ERROR HAS BEEN DETECTED IN THE DATA.
C-----------------------------------------------------------------------
C
      SUBROUTINE ERROR
      COMMON IOUT
      WRITE(IOUT,1)
1     FORMAT(44H ************ ERROR IN INPUT DATA **********)
      RETURN
      END
C
C-----------------------------------------------------------------------
C SUBROUTINE: REMEZ
C   THIS SUBROUTINE IMPLEMENTS THE REMEZ EXCHANGE ALGORITHM
C   FOR THE WEIGHTED CHEBYSHEV APPROXIMATION OF A CONTINUOUS
C   FUNCTION WITH A SUM OF COSINES.  INPUTS TO THE SUBROUTINE
C   ARE A DENSE GRID WHICH REPLACES THE FREQUENCY AXIS, THE
C   DESIRED FUNCTION ON THIS GRID, THE WEIGHT FUNCTION ON THE
C   GRID, THE NUMBER OF COSINES, AND AN INITIAL GUESS OF THE
C   EXTREMAL FREQUENCIES.  THE PROGRAM MINIMIZES THE CHEBYSHEV
C   ERROR BY DETERMINING THE BEST LOCATION OF THE EXTREMAL
C   FREQUENCIES (POINTS OF MAXIMUM ERROR) AND THEN CALCULATES
C   THE COEFFICIENTS OF THE BEST APPROXIMATION.
C-----------------------------------------------------------------------
```

```
      COMMON NITER,IOUT
      DIMENSION IEXT(34),AD(34),ALPHA(34),X(34),Y(34)
      DIMENSION DES(544),GRID(544),WT(544)
      DIMENSION A(34),P(65),Q(65)
      DOUBLE PRECISION PI2,DNUM,DDEN,DTEMP,A,P,Q
      DOUBLE PRECISION DK,DAK
      DOUBLE PRECISION AD,DEV,X,Y
      DOUBLE PRECISION GEE,D
C
C  THE PROGRAM ALLOWS A MAXIMUM NUMBER OF ITERATIONS OF 25
C
      ITRMAX=25
      DEVL=-1.0
      NZ=NFCNS+1
      NZZ=NFCNS+2
      NITER=0
100   CONTINUE
      IEXT(NZZ)=NGRID+1
      NITER=NITER+1
      IF(NITER.GT.ITRMAX) GO TO 400
      DO 110 J=1,NZ
      JXT=IEXT(J)
      DTEMP=GRID(JXT)
      DTEMP=DCOS(DTEMP*PI2)
110   X(J)=DTEMP
      JET=(NFCNS-1)/15+1
      DO 120 J=1,NZ
120   AD(J)=D(J,NZ,JET)
      DNUM=0.0
      DDEN=0.0
      K=1
      DO 130 J=1,NZ
      L=IEXT(J)
      DTEMP=AD(J)*DES(L)
      DNUM=DNUM+DTEMP
      DTEMP=FLOAT(K)*AD(J)/WT(L)
      DDEN=DDEN+DTEMP
130   K=-K
      DEV=DNUM/DDEN
      WRITE(IOUT,131) DEV
131   FORMAT(1X,12HDEVIATION = ,F12.9)
      NU=1
      IF(DEV.GT.0.0) NU=-1
      DEV=-FLOAT(NU)*DEV
      K=NU
      DO 140 J=1,NZ
      L=IEXT(J)
      DTEMP=FLOAT(K)*DEV/WT(L)
```

```
      Y(J)=DES(L)+DTEMP
140   K=-K
      IF(DEV.GT.DEVL) GO TO 150
      CALL OUCH
      GO TO 400
150   DEVL=DEV
      JCHNGE=0
      K1=IEXT(1)
      KNZ=IEXT(NZ)
      KLOW=0
      NUT=-NU
      J=1
C
C  SEARCH FOR THE EXTREMAL FREQUENCIES OF THE BEST
C  APPROXIMATION
C
200   IF(J.EQ.NZZ) YNZ=COMP
      IF(J.GE.NZZ) GO TO 300
      KUP=IEXT(J+1)
      L=IEXT(J)+1
      NUT=-NUT
      IF(J.EQ.2) Y1=COMP
      COMP=DEV
      IF(L.GE.KUP) GO TO 220
      ERR=GEE(L,NZ)
      ERR=(ERR-DES(L))*WT(L)
      DTEMP=FLOAT(NUT)*ERR-COMP
      IF(DTEMP.LE.0.0) GO TO 220
      COMP=FLOAT(NUT)*ERR
210   L=L+1
      IF(L.GE.KUP) GO TO 215
      ERR=GEE(L,NZ)
      ERR=(ERR-DES(L))*WT(L)
      DTEMP=FLOAT(NUT)*ERR-COMP
      IF(DTEMP.LE.0.0) GO TO 215
      COMP=FLOAT(NUT)*ERR
      GO TO 210
215   IEXT(J)=L-1
      J=J+1
      KLOW=L-1
      JCHNGE=JCHNGE+1
      GO TO 200
220   L=L-1
225   L=L-1
      IF(L.LE.KLOW) GO TO 250
      ERR=GEE(L,NZ)
      ERR=(ERR-DES(L))*WT(L)
      DTEMP=FLOAT(NUT)*ERR-COMP
      IF(DTEMP.GT.0.0) GO TO 230
      IF(JCHNGE.LE.0) GO TO 225
      GO TO 260
```

```
230   COMP=FLOAT(NUT)*ERR
235   L=L-1
      IF(L.LE.KLOW) GO TO 240
      ERR=GEE(L,NZ)
      ERR=(ERR-DES(L))*WT(L)
      DTEMP=FLOAT(NUT)*ERR-COMP
      IF(DTEMP.LE.0.0) GO TO 240
      COMP=FLOAT(NUT)*ERR
      GO TO 235
240   KLOW=IEXT(J)
      IEXT(J)=L+1
      J=J+1
      JCHNGE=JCHNGE+1
      GO TO 200
250   L=IEXT(J)+1
      IF(JCHNGE.GT.0) GO TO 215
255   L=L+1
      IF(L.GE.KUP) GO TO 260
      ERR=GEE(L,NZ)
      ERR=(ERR-DES(L))*WT(L)
      DTEMP=FLOAT(NUT)*ERR-COMP
      IF(DTEMP.LE.0.0) GO TO 255
      COMP=FLOAT(NUT)*ERR
      GO TO 210
260   KLOW=IEXT(J)
      J=J+1
      GO TO 200
300   IF(J.GT.NZZ) GO TO 320
      IF(K1.GT.IEXT(1)) K1=IEXT(1)
      IF(KNZ.LT.IEXT(NZ)) KNZ=IEXT(NZ)
      NUT1=NUT
      NUT=-NU
      L=0
      KUP=K1
      COMP=YNZ*(1.00001)
      LUCK=1
310   L=L+1
      IF(L.GE.KUP) GO TO 315
      ERR=GEE(L,NZ)
      ERR=(ERR-DES(L))*WT(L)
      DTEMP=FLOAT(NUT)*ERR-COMP
      IF(DTEMP.LE.0.0) GO TO 310
      COMP=FLOAT(NUT)*ERR
      J=NZZ
      GO TO 210
315   LUCK=6
      GO TO 325
320   IF(LUCK.GT.9) GO TO 350
      IF(COMP.GT.Y1) Y1=COMP
      K1=IEXT(NZZ)
325   L=NGRID+1
```

```
      KLOW=KNZ
      NUT=-NUT1
      COMP=Y1*(1.00001)
330   L=L-1
      IF(L.LE.KLOW) GO TO 340
      ERR=GEE(L,NZ)
      ERR=(ERR-DES(L))*WT(L)
      DTEMP=FLOAT(NUT)*ERR-COMP
      IF(DTEMP.LE.0.0) GO TO 330
      J=NZZ
      COMP=FLOAT(NUT)*ERR
      LUCK=LUCK+10
      GO TO 235
340   IF(LUCK.EQ.6) GO TO 370
      DO 345 J=1,NFCNS
      NZZMJ=NZZ-J
      NZMJ=NZ-J
345   IEXT(NZZMJ)=IEXT(NZMJ)
      IEXT(1)=K1
      GO TO 100
350   KN=IEXT(NZZ)
      DO 360 J=1,NFCNS
360   IEXT(J)=IEXT(J+1)
      IEXT(NZ)=KN
      GO TO 100
370   IF(JCHNGE.GT.0) GO TO 100
C
C  CALCULATION OF THE COEFFICIENTS OF THE BEST APPROXIMATION
C  USING THE INVERSE DISCRETE FOURIER TRANSFORM
C
400   CONTINUE
      NM1=NFCNS-1
      FSH=1.0E-06
      GTEMP=GRID(1)
      X(NZZ)=-2.0
      CN=2*NFCNS-1
      DELF=1.0/CN
      L=1
      KKK=0
      IF(GRID(1).LT.0.01.AND.GRID(NGRID).GT.0.49) KKK=1
      IF(NFCNS.LE.3) KKK=1
      IF(KKK.EQ.1) GO TO 405
      DTEMP=DCOS(PI2*GRID(1))
      DNUM=DCOS(PI2*GRID(NGRID))
      AA=2.0/(DTEMP-DNUM)
      BB=-(DTEMP+DNUM)/(DTEMP-DNUM)
405   CONTINUE
      DO 430 J=1,NFCNS
      FT=FLOAT(J-1)
      FT=FT*DELF
      XT=DCOS(PI2*FT)
```

```
      IF(KKK.EQ.1) GO TO 410
      XT=(XT-BB)/AA
      XT1=SQRT(1.0-XT*XT)
      FT=ATAN2(XT1,XT)/PI2
410   XE=X(L)
      IF(XT.GT.XE) GO TO 420
      IF((XE-XT).LT.FSH) GO TO 415
      L=L+1
      GO TO 410
415   A(J)=Y(L)
      GO TO 425
420   IF((XT-XE).LT.FSH) GO TO 415
      GRID(1)=FT
      A(J)=GEE(1,NZ)
425   CONTINUE
      IF(L.GT.1) L=L-1
430   CONTINUE
      GRID(1)=GTEMP
      DDEN=PI2/CN
      DO 510 J=1,NFCNS
      DTEMP=0.0
      DNUM=J-1
      DNUM=DNUM*DDEN
      IF(NM1.LT.1) GO TO 505
      DO 500 K=1,NM1
      DAK=A(K+1)
      DK=K
500   DTEMP=DTEMP+DAK*DCOS(DNUM*DK)
505   DTEMP=2.0*DTEMP+A(1)
510   ALPHA(J)=DTEMP
      DO 550 J=2,NFCNS
550   ALPHA(J)=2.0*ALPHA(J)/CN
      ALPHA(1)=ALPHA(1)/CN
      IF(KKK.EQ.1) GO TO 545
      P(1)=2.0*ALPHA(NFCNS)*BB+ALPHA(NM1)
      P(2)=2.0*AA*ALPHA(NFCNS)
      Q(1)=ALPHA(NFCNS-2)-ALPHA(NFCNS)
      DO 540 J=2,NM1
      IF(J.LT.NM1) GO TO 515
      AA=0.5*AA
      BB=0.5*BB
515   CONTINUE
      P(J+1)=0.0
      DO 520 K=1,J
      A(K)=P(K)
520   P(K)=2.0*BB*A(K)
      P(2)=P(2)+A(1)*2.0*AA
      JM1=J-1
      DO 525 K=1,JM1
525   P(K)=P(K)+Q(K)+AA*A(K+1)
      JP1=J+1
```

```
      DO 530 K=3,JP1
530   P(K)=P(K)+AA*A(K-1)
      IF(J.EQ.NM1) GO TO 540
      DO 535 K=1,J
535   Q(K)=-A(K)
      NF1J=NFCNS-1-J
      Q(1)=Q(1)+ALPHA(NF1J)
540   CONTINUE
      DO 543 J=1,NFCNS
543   ALPHA(J)=P(J)
545   CONTINUE
      IF(NFCNS.GT.3) RETURN
      ALPHA(NFCNS+1)=0.0
      ALPHA(NFCNS+2)=0.0
      RETURN
      END
C
C-----------------------------------------------------------------------
C FUNCTION: D
C   FUNCTION TO CALCULATE THE LAGRANGE INTERPOLATION
C   COEFFICIENTS FOR USE IN THE FUNCTION GEE.
C-----------------------------------------------------------------------
C
      DOUBLE PRECISION FUNCTION D(K,N,M)
      COMMON PI2,AD,DEV,X,Y,GRID,DES,WT,ALPHA,IEXT,NFCNS,NGRID
      DIMENSION IEXT(34),AD(34),ALPHA(34),X(34),Y(34)
      DIMENSION DES(544),GRID(544),WT(544)
      DOUBLE PRECISION AD,DEV,X,Y
      DOUBLE PRECISION Q
      DOUBLE PRECISION PI2
      D=1.0
      Q=X(K)
      DO 3 L=1,M
      DO 2 J=L,N,M
      IF(J-K)1,2,1
1     D=2.0*D*(Q-X(J))
2     CONTINUE
3     CONTINUE
      D=1.0/D
      RETURN
      END
C
C-----------------------------------------------------------------------
C FUNCTION:GEE
C   FUNCTION TO EVALUATE THE FREQUENCY RESPONSE USING THE
C   LAGRANGE INTERPOLATION FORMULA IN THE BARYCENTRIC FORM.
C-----------------------------------------------------------------------
C
      DOUBLE PRECISION FUNCTION GEE(K,N)
      COMMON PI2,AD,DEV,X,Y,GRID,DES,WT,ALPHA,IEXT,NFCNS,NGRID
      DIMENSION IEXT(34),AD(34),ALPHA(34),X(34),Y(34)
```

```
      DIMENSION DES(544),GRID(544),WT(544)
      DOUBLE PRECISION P,C,D,XF
      DOUBLE PRECISION PI2
      DOUBLE PRECISION AD,DEV,X,Y
      P=0.0
      XF=GRID(K)
      XF=DCOS(PI2*XF)
      D=0.0
      DO 1 J=1,N
      C=XF-X(J)
      C=AD(J)/C
      D=D+C
1     P=P+C*Y(J)
      GEE=P/D
      RETURN
      END
C
C-----------------------------------------------------------------------
C SUBROUTINE: OUCH
C   WRITES AN ERROR MESSAGE WHEN THE ALGORITHM FAILS TO
C   CONVERGE.  THERE SEEM TO BE TWO CONDITIONS UNDER WHICH
C   THE ALGORITHM FAILS TO CONVERGE: (1) THE INITIAL
C   GUESS FOR THE EXTREMAL FREQUENCIES IS SO POOR THAT
C   THE EXCHANGE ITERATION CANNOT GET STARTED, OR
C   (2) NEAR THE TERMINATION OF A CORRECT DESIGN,
C   THE DEVIATION DECREASES DUE TO ROUNDING ERRORS
C   AND THE PROGRAM STOPS.  IN THIS LATTER CASE THE
C   FILTER DESIGN IS PROBABLE ACCEPTABLE, BUT SHOULD
C   BE CHECKED BY COMPUTING A FREQUENCY RESPONSE.
C-----------------------------------------------------------------------
C
      SUBROUTINE OUCH
      COMMON NITER,IOUT
      WRITE(IOUT,1)NITER
1     FORMAT(44H ************ FAILURE TO CONVERGE **********,3(/),
     140HPROBABLE CAUSE IS MACHINE ROUNDING ERROR,3(/),
     222HNUMBER OF ITERATIONS =,I4,3(/),
     338HIF THE NUMBER OF ITERATIONS EXCEEDS 3,3(/),
     461H THE DESIGN MAY BE CORRECT, BUT SHOULD BE VERIFIED WITH AN FFT)
      RETURN
      END
```

FIRF2 Output

```
DEVIATION =   .000734616
DEVIATION =   .006314959
DEVIATION =   .021566622
DEVIATION =   .026202958
DEVIATION =  -.032680455
DEVIATION =  -.034435450
DEVIATION =  -.034448376
DEVIATION =  -.034448590
```

```
**************************************************************************

                    BANDPASS FILTER

                  FILTER LENGTH =  55

              ***** IMPULSE RESPONSE *****

          H( 1) =    .10662768E-02 = H( 55)
          H( 2) =    .63777557E-02 = H( 54)
          H( 3) =    .35755956E-02 = H( 53)
          H( 4) =   -.90678427E-02 = H( 52)
          H( 5) =   -.90907477E-02 = H( 51)
          H( 6) =    .29156702E-02 = H( 50)
          H( 7) =    .39638160E-02 = H( 49)
          H( 8) =    .11171967E-01 = H( 48)
          H( 9) =    .11646770E-01 = H( 47)
          H(10) =   -.99630672E-02 = H( 46)
          H(11) =   -.92384685E-02 = H( 45)
          H(12) =   -.20406403E-01 = H( 44)
          H(13) =   -.19460432E-01 = H( 43)
          H(14) =    .31243082E-01 = H( 42)
          H(15) =    .63045118E-02 = H( 41)
          H(16) =   -.20482879E-01 = H( 40)
          H(17) =    .65741474E-02 = H( 39)
          H(18) =   -.11201589E-02 = H( 38)
          H(19) =    .41956894E-01 = H( 37)
          H(20) =    .35784297E-01 = H( 36)
          H(21) =    .34744851E-01 = H( 35)
          H(22) =    .71496233E-01 = H( 34)
          H(23) =   -.17138839E+00 = H( 33)
          H(24) =   -.18255037E+00 = H( 32)
          H(25) =    .74059084E-01 = H( 31)
          H(26) =   -.10317425E+00 = H( 30)
          H(27) =    .25716718E-01 = H( 29)
          H(28) =    .37813556E+00 = H( 28)
```

	BAND 1	BAND 2	BAND 3	BAND 4
LOWER BAND EDGE	.0000000	.1000000	.1800000	.3000000
UPPER BAND EDGE	.0500000	.1500000	.2500000	.3600000
DESIRED VALUE	.0000000	1.0000000	.0000000	1.0000000
WEIGHTING	10.0000000	1.0000000	3.0000000	1.0000000
DEVIATION	.0034449	.0344486	.0114829	.0344486
DEVIATION IN DB	-49.2565765	.2941777	-38.7990036	.2941777

	BAND 5
LOWER BAND EDGE	.4100000
UPPER BAND EDGE	.5000000
DESIRED VALUE	.0000000
WEIGHTING	20.0000000
DEVIATION	.0017224
DEVIATION IN DB	-55.2771683

EXTREMAL FREQUENCIES--MAXIMA OF THE ERROR CURVE

.0000000	.0167411	.0323661	.0446429	.0500000
.1000000	.1089285	.1267856	.1424105	.1500000
.1800000	.1855803	.1978571	.2134820	.2302230
.2436158	.2500000	.3000000	.3122771	.3323668
.3502244	.3600000	.4100000	.4155805	.4289737
.4457151	.4635727	.4814302	.5000000	

****

AD-A118 066 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 17/2
A GENERAL PURPOSE MINI-COMPUTER BASED DIGITAL SIGNAL PROCESSING--ETC(U)
JUN 82 W TODD
UNCLASSIFIED AFIT/GE/EE/82J-14 NL

2 OF 2
AD A
118066

END
DATE
FILMED
09-82
DTIC

A

06

```
FTN4,L
C
C
      PROGRAM COVAU
C
C---------------------------------------------------------------
C   MAIN PROGRAM:  TEST PROGRAM FOR COVAR AND AUTO
C   AUTHORS:       A. H. GRAY, JR. AND J. D. MARKEL
C      GRAY -      UNIV. OF CALIF., SANTA BARBARA, CA 93109
C      BOTH -      SIGNAL TECHNOLOGY, INC., 15 W. DE LA GUERRA
C                  STREET, SANTA BRABARA, CA  93101
C---------------------------------------------------------------
C
      DIMENSION X(100), A(21), RC(21)
C
      IOUTD = 1
      X(1) = 1.
      X(2) = 1.44
      X(3) = 1.44*X(2) -1.26
      DO 10 J=4,40
        X(J) = 1.44*X(J-1) - 1.26*X(J-2) + .81*X(J-3)
10    CONTINUE
9999  FORMAT(I3,2E15.6)
9998  FORMAT(E12.6/)
      N = 40
      DO 20 M=2,3
        CALL COVAR(N,X,M,A,ALPHA,RC)
        MP = M+1
        WRITE(IOUTD,9999)  (I,A(I),RC(I),I=1,MP)
        WRITE(IOUTD,9998) ALPHA
        CALL AUTO(N,X,M,A,ALPHA,RC)
        WRITE(IOUTD,9999)  (I,A(I),RC(I),I=1,MP)
        WRITE(IOUTD,9998) ALPHA
20    CONTINUE
      DO 30 JB=1,40
        J = 44-JB
        X(J) = X(J-3)
30    CONTINUE
      DO 40 J=1,3
        JB = 43+J
        X(J) = 0.
        X(JB) = 0.
40    CONTINUE
      M=3
      N=46
      CALL COVAR(N,X,M,A,ALPHA,RC)
      MP = M+1
      WRITE(IOUTD,9999)  (I,A(I),RC(I),I=1,MP)
      WRITE(IOUTD,9998) ALPHA
      CALL AUTO(N,X,M,A,ALPHA,RC)
      WRITE(IOUTD,9999)  (I,A(I),RC(I),I=1,MP)
      WRITE(IOUTD,9998) ALPHA
      STOP
      END
C
```

```
C
C-----------------------------------------------------------------
C SUBROUTINE: COVAR
C A SUBROUTINE FOR IMPLEMENTING THE COVARIANCE
C METHOD OF LINEAR PREDICTION ANALYSIS
C-----------------------------------------------------------------
C
      SUBROUTINE COVAR(N,X,M,A,ALPHA,GRC)
C
C   INPUTS:  N - NO. OF DATA POINTS
C            X(N) - INPUT DATA SEQUENCE
C            M - ORDER OF FILTER (M<21,SEE NOTE*)
C  OUTPUTS:  A - FILTER COEFFICIENTS
C            ALPHA - RESIDUAL "ENERGY".
C            GRC - "GENERALIZED REFLECTION COEFFICIENTS"
C
C *PROGRAM LIMITED TO M=20 BECAUSE OF THE DIMENSIONS
C B(M*(M-1)/2), BETA(M), AND CC(M+1)
C
      DIMENSION X(1),A(1),GRC(1)
      DIMENSION B(190),BETA(20),CC(21)
      MP = M+1
      MT = (MP*M)/2
      DO 10 J=1,MT
        B(J) = 0.
10    CONTINUE
      ALPHA = 0.
      CC(1) = 0.
      CC(2) = 0.
      DO 20 NP=MP,N
        NP1 = NP-1
        ALPHA = ALPHA+X(NP)*X(NP)
        CC(1) = CC(1)+X(NP)*X(NP1)
        CC(2) = CC(2)+X(NP1)*X(NP1)
20    CONTINUE
      B(1) = 1.
      BETA(1) = CC(2)
      GRC(1) = -CC(1)/CC(2)
      A(1) = 1.
      A(2) = GRC(1)
      ALPHA = ALPHA+GRC(1)*CC(1)
      MF = M
      DO 130 MINC=2,MF
        DO 30 J=1,MINC
          JP = MINC+2-J
          N1 = MP+1-JP
          N2 = N+1-MINC
          N3 = N+2-JP
          N4 = MP-MINC
          CC(JP) = CC(JP-1)+X(N4)*X(N1)-X(N2)*X(N3)
30      CONTINUE
        CC(1) = 0.
        DO 40 NP=MP,N
          N1 = NP-MINC
          CC(1) = CC(1)+X(N1)*X(NP)
```

```
40      CONTINUE
        MSUB = (MINC*MINC-MINC)/2
        MM1 = MINC-1
        MM1 = MINC-1
        N1 = MSUB+MINC
        B(N1) = 1.
        DO 80 IP=1,MM1
          ISUB = (IP*IP-IP)/2
          IF(BETA(IP)) 150,150,50
50        GAM = 0.
          DO 60 J=1,IP
            N1 = ISUB+J
            GAM = GAM+CC(J+1)*B(N1)
60        CONTINUE
          GAM = GAM/BETA(IP)
          DO 70 JP=1,IP
            N1 = MSUB+JP
            N2 = ISUB+JP
            B(N1) = B(N1)-GAM*B(N2)
70        CONTINUE
80      CONTINUE
        BETA(MINC) = 0.
        DO 90 J=1,MINC
          N1 = MSUB+J
          BETA(MINC) = BETA(MINC)+CC(J+1)*B(N1)
90      CONTINUE
        IF(BETA(MINC)) 150,150,100
100     S=0.
        DO 110 IP=1,MINC
          S = S+CC(IP)*A(IP)
110     CONTINUE
        GRC(MINC) = -S/BETA(MINC)
        DO 120 IP=2,MINC
          M2 = MSUB+IP-1
          A(IP) = A(IP)+GRC(MINC)*B(M2)
120     CONTINUE
        A(MINC+1) = GRC(MINC)
        S = GRC(MINC)*GRC(MINC)*BETA(MINC)
        ALPHA = ALPHA-S
        IF(ALPHA) 150,150,130
130   CONTINUE
140   RETURN
150   CONTINUE
C
C WARNING - SINGULAR MATRIX
C
      IOUTD = 1
      WRITE(IOUTD,9999)
9999  FORMAT(34H WARNING - SINGULAR MATRIX - COVAR)
      GO TO 140
      END
C
C
C
C--------------------------------------------------------------
C SUBROUTINE: AUTO
```

```
C A SUBROUTINE FOR IMPLEMENTING THE AUTOCORRELATION
C METHOD OF LINEAR PREDICTION ANALYSIS
C-------------------------------------------------------------
C
      SUBROUTINE AUTO(N,X,M,A,ALPHA,RC)
C
C   INPUTS:  N - NO. OF DATA POINTS
C            X(N) - INPUT DATA SEQUENCE
C            M - ORDER OF FILTER (M<21, SEE NOTE*)
C  OUTPUTS:  A - FILTER COEFFICIENTS
C            ALPHA - RESIDUAL "ENERGY
C            RC - REFLECTION COEFFICIENTS
C
C *PROGRAM LIMITED TO M<21 BECAUSE OF DIMENSIONS OF R(.)
C
      DIMENSION X(1),A(1),RC(1)
      DIMENSION R(21)
C
      MP = M+1
      DO 20 K=1,MP
        R(K) = 0.
        NK = N-K+1
        DO 10 NP=1,NK
          N1 = NP+K-1
          R(K) = R(K)+X(NP)*X(N1)
10      CONTINUE
20    CONTINUE
      RC(1) = -R(2)/R(1)
      A(1) = 1.
      A(2) = RC(1)
      ALPHA = R(1)+R(2)*RC(1)
      DO 50 MINC=2,M
        S=0.
        DO 30 IP=1,MINC
          N1 = MINC-IP+2
          S = S+R(N1)*A(IP)
30      CONTINUE
        RC(MINC) = -S/ALPHA
        MH = MINC/2+1
        DO 40 IP=2,MH
          IB = MINC-IP+2
          AT = A(IP)+RC(MINC)*A(IB)
          A(IB) = A(IB)+RC(MINC)*A(IP)
          A(IP) = AT
40      CONTINUE
        A(MINC+1) = RC(MINC)
        ALPHA = ALPHA+RC(MINC)*S
        IF(ALPHA) 70,70,50
50    CONTINUE
60    RETURN
70    CONTINUE
C
C WARNING - SINGULAR MATRIX
C
      IOUTD = 1
```

```
      WRITE(IOUTD,9999)
9999  FORMAT(33H WARINING - SINGULAR MATRIX - AUTO)
      GO TO 60
      END
```

COVAR Output

```
 1     .100000E+01    -.891117E+00
 2    -.108020E+01     .197714E+00
 3     .197714E+00     .000000E+00
.172096E+01

 1     .100000E+01    -.905549E+00
 2    -.116200E+01     .283199E+00
 3     .283198E+00     .000000E+00
.308389E+01

 1     .100000E+01    -.935487E+00
 2    -.144000E+01     .313708E+00
 3     .125999E+01    -.809995E+00
 4    -.809995E+00     .000000E+00
.162125E-04

 1     .100000E+01    -.905549E+00
 2    -.136308E+01     .283199E+00
 3     .110826E+01    -.710039E+00
 4    -.710038E+00     .000000E+00
.152913E+01

 1     .100000E+01    -.905549E+00
 2    -.136308E+01     .283199E+00
 3     .110826E+01    -.710039E+00
 4    -.710039E+00     .000000E+00
.152913E+01

 1     .100000E+01    -.905549E+00
 2    -.136308E+01     .283198E+00
 3     .110826E+01    -.710038E+00
 4    -.710038E+00     .000000E+00
.152913E+01
```

```
FTN4,L
C
C
      PROGRAM(),FWIIR NAMED COMMON
C
C-----------------------------------------------------------------
C  MAIN PROGRAM:  FWIIR
C  AUTHORS:       KENNETH STEIGLITZ AND BRUCE D. LADENDORF
C                 PRINCETON UNIVERSITY, PRINCETON, NJ 08540
C                 VERSION OCTOBER 15, 1978
C
C  INPUT:         A GRID OF FREQUENCY POINTS AND A SET OF
C                 (ESSENTIALLY) INFINITE PRECISION COEF-
C                 FICIENTS. SEE SUBROUTINE INDAT FOR
C                 DETAILED INPUT INFORMATION.
C
C  THIS PROGRAM DESIGNS FINITE WORD-LENGTH IIR DIGITAL
C  FILTERS.  THE METHOD USES RANDOMIZED PATTERN SEARCH
C  OF HOOKE AND JEEVES, AND IS DESCRIBED IN "DESIGNING
C  SHORT-WORD RECURSIVE DIGITAL FILTERS," BY KENNETH
C  STEIGLITZ, IN PROC. 9TH ANNUAL ALLERTON CONF. ON CIRCUIT
C  AND SYSTEM TH., PP.778-788  OCT. 1971.  REPRINTED IN
C  DIGITAL SIGNAL PROCESSING II, EDITED BY THE DIGITAL
C  SIGNAL PROCESSING COMMITTEE OF THE GROUP ON ASSP, IEEE
C  PRESS, N. Y., 1976.
C-----------------------------------------------------------------
C
      DOUBLE PRECISION W(80),Y(80),WEIGHT(80),X(36)
      DOUBLE PRECISION SDELTA,DELTAZ,DUMMY,RHO,EST,F
      LOGICAL PRINT,TWOPT,FIXCOF
      COMMON/ABC/X,NSET,NTWOPT,NBIT1,NBIT2,NSECT,N
      COMMON/RAW/W,Y,WEIGHT,M,KOUNT
      COMMON/STEER/PRINT,TWOPT,FIXCOF
      COMMON/MACH/IND,IOUTD
C
C SET UP MACHINE CONSTANTS
C
      WRITE(1,100)
100   FORMAT(/"WHAT IS THE INPUT LU#?")
      READ(1,*) IND
      WRITE(1,200)
200   FORMAT(/"WHAT IS THE OUTPUT LU#?")
      READ(1,*) IOUTD
C
C READ IN PROBLEM DATA
C
      CALL INDAT
C
C SET SMALL AND LARGE DELTA
C
      SDELTA = (.5D0)**NBIT1
      DELTAZ = (.5D0)**NBIT2
C
C SET RANDOM NUMBER GENERATOR BY CALLING IT NSET TIMES
C
```

```
      DO 10 J=1,NSET
        DUMMY = DBLE(UNI(0))
10    CONTINUE
C
C TURN OFF THE PRINTING OPTION FOR FUNCT
C
      PRINT = .FALSE.
C
C ROUND OFF THE COEFFICIENTS TO NBIT1 BITS
C
      DO 20 J=1,N
        X(J) = DSIGN(SDELTA*FLOAT(IDINT(DABS(X(J))/SDELTA
     +      +.5D0)),X(J))
20    CONTINUE
      WRITE(IOUTD,9999) NBIT1
9999  FORMAT(//37HTHE INITIAL COEFFICIENTS, ROUNDED TO ,I3,
     +      10H BITS, ARE)
      DO 30 J=1,NSECT
        WRITE(IOUTD,9998) J,X(4*J-3),X(4*J-2),X(4*J-1),X(4*J)
30    CONTINUE
9998  FORMAT(7HSECTION,I3/1H ,2D20.10/1H ,2D20.10)
C
C SET PARAMETER VALUES FOR H-AND-J; THESE MAY BE CHANGED IF
C DESIRED. H-AND-J MULTIPLIES THE STEP SIZE BY RHO; TAKE RHO=
C
      RHO = .5D0
C
C H-AND-J STOPS IF THE FUNCTION VALUE FALLS BELOW EST;
C TAKE EST = 0.
C
      EST = 0.
C
C H-AND-J STOPS IF WE EXCEED "LIMIT" FUNCT CALLS;
C TAKE LIMIT = 10,000
C
      LIMIT = 10000
      WRITE(IOUTD,9997)
9997  FORMAT(/47HNEXT FOLLOWS A REPORT FROM THE SEARCH ALGORI
      CALL HANDJ(N,DELTAZ,SDELTA,RHO,EST,LIMIT,X)
      WRITE(IOUTD,9996) NBIT1
9996  FORMAT(/30HTHE FINAL COEFFICIENTS, HAVING, I3,
     +      10H BITS, ARE)
      DO 40 J=1,NSECT
        WRITE(IOUTD,9998) J,X(4*J-3),X(4*J-2),X(4*J-1),X(4*J)
40    CONTINUE
C
C PRINT OUT THE FINAL FINE GRID AND QUIT
C
      PRINT = .TRUE.
      CALL FUNCT(N,X,F)
      STOP
      END
C
C-----------------------------------------------------------
C SUBROUTINE:  INDAT
```

```
C THIS SUBROUTINE READS IN THE PROBLEM DATA:
C CARD 1 HAS THE NUMBER OF COEFFICIENTS PER STAGE (3 OR 4),
C THE INITIAL SETTING OF THE RANDOM NUMBER GENERATOR,
C WHETHER TWO-OPT SHOULD BE USED (0 IS NO, 1 IS YES), THE
C DESIRED BIT LENGTH OF THE COEFFICIENTS, AND THE INITIAL
C SEARCH DELTA (IN BITS) FOR THE HOOKE AND JEEVES ALGORITHM.
C
C THE NEXT SET OF CARDS SPECIFIES THE GRID, ONE GRID POINT
C PER CARD:  EACH CARD HAS FIRST THE FREQUENCY IN FRACTIONS
C OF THE NYQUIST FREQUENCY, NEXT THE DESIRED TRANSFER FUNC-
C TION, AND THIRD THE TOLERANCE. TO INDICATE THE END OF THE
C GRID POINTS, SPECIFY A FREQUENCY OF 1. THE M CARDS ARE
C COUNTED, M.LE.100. THE PRESENT SUBROUTINE FUNCT ASSUMES FOR
C THE A CALCULATION`THAT THE DESIRED TRANSFER FUNCTION TAKES
C ON ONLY THE VALUES 0 OR 1. THAT IS, THE FILTER HAS ONLY PASS
C AND STOP BANDS, AND EVERY PASS BAND HAS DESIRED VALUE OF 1,
C AND THE SAME TOLERANCE.
C
C THE NEXT CARD HAS THE NUMBER OF SECOND-ORDER SECTIONS, NSECT.
C
C THE NEXT CARDS HAVE THE "INFINITE PRECISION" COEFFICIENTS,
C ONE PER CARD. THE COEFFICIENTS FOR SECTION 1 ARE GIVEN
C FIRST, NEXT SECTION 2, ETC. WHITHIN EACH SECTION THEY ARE
C IN THE ORDER: NUMERATOR COEFFICIENT OF Z**(-1), NUMERATOR
C COEFFICIENT OF Z**(-2) (IF 4 PER SECTION), DENOMINATOR
C COEFFICIENT OF Z**(-1), DENOMINATOR COEFFICIENT OF Z**(-2).
C THERE ARE 3*NSECT OR 4*NSECT COEFFICIENTS READ IN, DEPEN-
C DING ON WHETHER THERE ARE 3 OR 4 COEFFICIENTS PER SECTION,
C NSECT.LE.9. IN THE CASE OF 3 COEFFICIENTS PER SECTION, THE
C SECOND NUMBERATOR COEFFICIENT IS SET TO 1 AND FROZEN.
C------------------------------------------------------------
C
      SUBROUTINE INDAT
      DOUBLE PRECISION W(80),Y(80),WEIGHT(80),X(36)
      LOGICAL PRINT,TWOPT,FIXCOF
      COMMON/ABC/X,NSET,NTWOPT,NBIT1,NBITS,NSECT,N
      COMMON/RAW/W,Y,WEIGHT,M,KOUNT
      COMMON/ITEER/PRINT,TWOPT,FIXCOF
      COMMON/MACH/IND,IOUTD
C
C INITIALIZE KOUNT, THE NUMBER OF FUNCTION EVALUATIONS
C
      KOUNT = 0
C
C READ AND WRITE CARD 1 PARAMETERS
C
      READ(IND,*) NCOEF,NSET,NTWOPT,NBIT1,NBITS
      WRITE(IOUTD,9999) NCOEF,NSET,NTWOPT,NBIT1,NBITS
9999  FORMAT(//23H ***** INPUT DATA *****/9HNUMBER OF
     + 16H COEFFICIENTS PE,10HR STAGE IS, I3/
     + 25HRANDOM INITIALIZATION IS ,I3/9HTWOPT IS,I3/
     + 43HFINAL ( AND INITIAL ROUNDING ) PRECISION IS,
     + I3,5H BITS/23HINITIAL SEARCH DELTA IS,I3,5H BITS)
C
```

```
C SET LOGICAL VARIABLE TWOPT IF TWO-OPT IS CALLED FOR
C
      TWOPT = .FALSE.
      IF(NTWOPT.EQ.1) TWOPT = .TRUE.
C
C READ AND WRITE THE GRID
C
      WRITE(IOUTD,9997)
9997  FORMAT(/19HGRID SPECIFICATIONS/
     +         29HPOINT NO.             FREQUENCY,
     +         40H              DESIRED Y             TOLERANCE)
      M = 0
10    M = M+1
      READ(IND,9996) W(M),Y(M),WEIGHT(M)
9996  FORMAT(3F10.6)
      WRITE(IOUTD,9995) M,W(M),Y(M),WEIGHT(M)
9995  FORMAT(I9,3D20.10)
      WEIGHT(M) = 1.D0/WEIGHT(M)
      IF(W(M).LT.1.D0) GO TO 10
C
C READ AND WRITE THE COEFFICIENTS
C
      READ(IND,*) NSECT
      WRITE(IOUTD,9994) NSECT
9994  FORMAT(/39HINITIAL HIGH PRECISION COEFFICIENTS FOR,
     *  I3,13H SECTIONS ARE)
      N = 4*NSECT
      IF(NCOEF.EQ.3) GO TO 20
      FIXCOF = .FALSE.
      READ(IND,9993) (X(J),J=1,N)
9993  FORMAT(F20.16)
      GO TO 40
20    FIXCOF = .TRUE.
      READ(IND,9993) (X(4*J-3),X(4*J-1),X(4*J),J=1,NSECT)
      DO 30 J=1,NSECT
        X(4*J-2) = 1.D0
30    CONTINUE
40    CONTINUE
      DO 50 J=1,NSECT
        WRITE(IOUTD,9992) J,X(4*J-3),X(4*J-2),X(4*J-1),X(4*J)
50    CONTINUE
9992  FORMAT(7HSECTION,I3/1H ,2D20.10/1H ,2D20.10)
      RETURN
      END
C
C-----------------------------------------------------------------
C SUBROUTINE:  FUNCT
C THIS COMPUTES THE VALUE OF THE ERROR FUNCTION, USING THE
C GRID. IF PRINT.EQ. .TRUE., IT COMPUTES THE ERROR FUNCTION
C AT A FINER GRID ( 10 POINTS FOR EACH GRID POINT), AND
C PRINTS THE RESULTS; THIS IS USED TO PRINT OUT THE FINAL
C TRANSFER FUNCTION.
C
C NOTE THAT THE CALCULATION OF THE CONSTANT MULTIPLIER A
C ASSUMES THAT ALL THE PASSBANDS HAVE THE SAME DESIRED VALUE
```

```
C OF 1. THIS MUST BE CHANGED TO A BAND-BY-BAND CALCULATION
C IF THERE IS MORE THAN ONE PASSBAND DESIRED VALUE.
C----------------------------------------------------------
C
      SUBROUTINE FUNCT(N,X,F)
      DOUBLE PRECISION W(80),Y(80),WEIGHT(80)
      DOUBLE PRECISION COST2(80),COS2T4(80)
      DOUBLE PRECISION SUB1(80),SUB2(80),SUB3(80)
      DOUBLE PRECISION SUB6(80),SUB7(80),SUB8(80)
      DOUBLE PRECISION X(36),YHT(80),E(80)
      DOUBLE PRECISION YHTHT(100,11)
      DOUBLE PRECISION A,F,PI,NUM,DEN,Y1,Y2,FREQ,CCC,SSS,A1,ERROR
      LOGICAL PRINT,TWOPT,FIXCOF
      COMMON/RAW/W,Y,WEIGHT,M,KOUNT
      COMMON/STEER/PRINT,TWOPT,FIXCOF
      COMMON/MACH/IND,IOUTD
C
C CHECK THAT THE POLES OF THE FILTER ARE INSIDE OR ON THE
C UNIT CIRCLE; NO CHECKS ARE MAKE ON NUMBERATOR COEFFICIENTS
C
      K = N/4
      DO 10 J=1,K
        J4 = J*4
        IF(X(J4).LE.1.D0.AND.1.D0+X(J4).GE.DABS(X(J4-1)))
     +     GO TO 10
        F = 1.D12
        RETURN
10    CONTINUE
      PI = 4.D0*DATAN(1.D0)
      IF(KOUNT.NE.0) GO TO 30
C
C COMPUTE AND SAVE THE TRIG FUNCTIONS AT THE GRID POINTS
C THE FIRST TIME THROUGH
C
      DO 20 I=1,M
        COST2(I) = 2.D0*DCOS(PI*W(I))
        COS2T4(I) = COST2(I)**2
20    CONTINUE
C
C MOVE THE CALCULATION OF CONSTANTS OUTSIDE THE INNER LOOP
C
30    DO 40 J=1,K
        J4 = J*4
        SUB1(J) = X(J4-3)**2+(X(J4-2)-1.D0)**2
        SUB2(J) = X(J4-3)*(X(J4-2)+1.D0)
        SUB3(J) = X(J4-2)
        SUB6(J) = X(J4-1)**2+(X(J4)-1.D0)**2
        SUB7(J) = X(J4-1)*(X(J4)+1.D0)
        SUB8(J) = X(J4)
40    CONTINUE
C
C EVALUATE THE MAGNITUDE OF THE TRANSFER FUNCTION, YHT,
C AT EACH OF THE M GRID POINTS
C
      DO 60 I=1,M
```

```
          NUM = 1.D0
          DEN = 1.D0
          DO 50 J=1,K
            NUM = NUM*(SUB1(J)+SUB2(J)*COS2T2(I)+SUB3(J)*COS2T4(I))
            DEN = DEN*(SUB6(J)+SUB7(J)*COS2T2(I)+SUB8(J)*COS2T4(I))
50        CONTINUE
          YHT(I) = DSQRT(NUM/DEN)
60      CONTINUE
C
C FIND THE LARGEST AND SMALLEST VALUES OF YHT IN THE
C PASSBANDS
C
        Y1 = 1.D12
        Y2 = 0.D0
        DO 70 I=1,M
          IF(Y(I).EQ.0.D0) GO TO 70
          Y1 = DMIN1(Y1,YHT(I))
          Y2 = DMAX1(Y2,YHT(I))
70      CONTINUE
C
C DEFINE THE CONSTANT MULTIPLIER A TO BE THE RECIPROCAL OF
C THE AVERAGE OF THE SMALLEST AND LARGEST YHT'S IN THE PASS-
C BANDS. THIS MAKES SENSE IF ALL THE PASSBANDS HAVE THE SAME
C DESIRED VALUE OF 1 AND THE SAME TOLERANCE; OTHERWISE THIS
C CALCULATION OF A MUST BE MODIFIED
C
        A = 2.D0/(Y1+Y2)
C
C CALCULATE THE WEIGHTED ERROR AT THE GRID POINTS
C
        F = 0.D0
        DO 80 I=1,M
          YHT(I) = A*YHT(I)
          E(I) = WEIGHT(I)*DABS(YHT(I)-Y(I))
          F = DMAX1(F,E(I))
80      CONTINUE
        KOUNT = KOUNT+1
        IF(.NOT.PRINT) RETURN
C
C IF REQUIRED, PRINT OUT THE RESULTS ON THE COARSE GRID
C
        WRITE(IOUTD,9999) A,F
9999    FORMAT(//"***** FINAL RESULTS *****"/17HTHE CONSTANT A IS,
     +  D20.10/32HFUNCTION VALUE ON COARSE GRID IS, D20.10)
        WRITE(IOUTD,9998)
9998    FORMAT(7X,"FREQUENCY",7X,9HDESIRED Y,8X,8HACTUAL Y,
     +  11X,5HERROR)
        DO 90 I=1,M
          WRITE(IOUTD,9997) W(I),Y(I),YHT(I),E(I)
90      CONTINUE
9997    FORMAT(4D16.8)
C
C TO RE-CALCULATE A, CALCULATE N/D=YHTHT ON THE FINE GRID
C THE SAME RESERVATION ABOUT THIS CALCULATION OF A APPLIES
C AS ABOVE
```

```
C
      Y1 = 1.D12
      Y2 = 0.D0
      MM = M-1
      DO 130 I=1,MM
C
C SKIP TRANSITION BANDS, DEFINED AS THOSE WHERE Y(I) CHANGES
C
        IF(Y(I).NE.Y(I+1)) GO TO 130
        DO 120 II=1,11
          IF(II.NE.11) GO TO 100
C
C FIND CASES WHERE THE LAST POINT OF THE BAND WILL NOT BE
C COMPUTED LATER
C
          IF(I.EQ.MM) GO TO 100
          IF(Y(I+1).NE.Y(I+2)) GO TO 100
          GO TO 120
100       FREQ = W(I)+(W(I+1)-W(I))*.1D0*FLOAT(II-1)
          CCC = 2.D0*DCOS(PI*FREQ)
          SSS = CCC**2
          NUM = 1.D0
          DEN = 1.D0
          DO 110 J=1,K
            J4 = J*4
            NUM = NUM*(X(J4-3)**2+(X(J4-2)-1.D0)**2+X(J4-3)*
     +             (X(J4-2)+1.D0)*CCC+X(J4-2)*SSS)
            DEN = DEN*(X(J4-1)**2+(X(J4)-1.D0)**2+X(J4-1)*
     +             (X(J4)+1.D0)*CCC+X(J4)*SSS)
110       CONTINUE
          YHTHT(I,II) = DSQRT(NUM/DEN)
          IF(Y(I).EQ.0.D0) GO TO 120
          Y1 = DMIN1(Y1,YHTHT(I,II))
          Y2 = DMAX1(Y2,YHTHT(I,II))
120     CONTINUE
130   CONTINUE
      A = 2.D0/(Y1+Y2)
C
C NOW REPEAT THE SAME LOOPS, FINDING THE ERROR AND PRINTING
C THE RESULTS
C
      WRITE(IOUTD,9996)
9996  FORMAT(//"FINAL RESULTS ON FINE GRID- 10 POINTS PER",
     +         11H GRID POINT)
      WRITE(IOUTD,9998)
      E = 0.D0
      DO 160 I=1,MM
        IF(Y(I).NE.Y(I+1)) GO TO 160
        IP = I+1
        WRITE(IOUTD,9995) I,IP
9995    FORMAT("***** BAND FROM GRID POINT",I5,3H TO,
     +         I5,6H *****)
        DO 150 II=1,11
          IF(II.NE.11) GO TO 140
          IF(I.EQ.MM) GO TO 140
```

```
          IF(Y(I+1).NE.Y(I+2)) GO TO 140
          GO TO 150
140       FREQ = W(I)+(W(I+1)-W(I))*.1D0*FLOAT(II-1)
          A1 = A*YHTHT(I,II)
          ERROR = WEIGHT(I)*DABS(A1-Y(I))
          F = DMAX1(F,ERROR)
          WRITE(IOUTD,9997) FREQ,Y(I),A1,ERROR
150     CONTINUE
160   CONTINUE
      WRITE(IOUTD,9994) A,F
9994  FORMAT(/"THE VALUE OF A FROM THE FINE GRID IS",D20.10//
     +43HTHE FINAL VALUE OF THE ERROR ON THE FINE GRID IS,D20.10)
      RETURN
      END
C
C-----------------------------------------------------------------
C SUBROUTINE:  HANDJ
C THIS IS AN IMPLEMENTATION OF A RANDOMIZED HOOKE AND JEEVES
C SEARCH ALGORITHM. IT FINDS LOCAL OPTIMA; THE COORDINATES ARE
C RANDOMLY ORDERED EACH TIME EXPLR IS INVOKED. FLOW CHARTS
C FOR THIS SUBROUTINE AND THE NEXT ARE INCLUDED IN THE
C REFERENCED PAPER.
C-----------------------------------------------------------------
C
      SUBROUTINE HANDJ(N,DELTAZ,IDELTA,RHO,FMIN,LIMIT,PSI)
      DOUBLE PRECISION PSI(36),THETA(36),PHI(36)
      DOUBLE PRECISION W(80),Y(80),WEIGHT(80)
      DOUBLE PRECISION FPSI,FPHI,DELTA,RHO,DELTAZ,IDELTA,FMIN
      COMMON/RAW/W,Y,WEIGHT,M,KOUNT
      COMMON/MACH/IND,IOUTD
      CALL FUNCT(N,PSI,FPSI)
      DELTA = DELTAZ
      WRITE(IOUTD,9999)
9999  FORMAT("NO. CALLS  PAT. SIZE",11X,"DELTA",7X,
     +  "OLD VALUE",7X,"NEW VALUE")
10    IF(DELTA.LT.IDELTA) GO TO 50
      IF(FPSI.LT.FMIN) GO TO 50
      IF(KOUNT.GT.LIMIT) GO TO 50
      CALL SET(PHI,PSI,N)
      FPHI = FPSI
      CALL EXPLR(PHI,FPHI,N,DELTA)
      I = 0
20    IF(FPHI.GE.FPSI) GO TO 40
      IF(FPSI.LT.FMIN) GO TO 40
      IF(KOUNT.GT.LIMIT) GO TO 40
      I = I+1
      WRITE(IOUTD,9998) KOUNT,I,DELTA,FPSI,FPHI
9998  FORMAT(3X,I6,5X,I6,3D16.8)
      CALL SET(THETA,PSI,N)
      CALL SET(PSI,PHI,N)
      FPSI = FPHI
      DO 30 J=1,N
        PHI(J) = 2.*PHI(J) - THETA(J)
30    CONTINUE
      CALL FUNCT(N,PHI,FPHI)
```

```
      CALL EXPLR(PHI,FPHI,N,DELTA)
      GO TO 20
40    IF(I.GT.0) GO TO 10
      DELTA = RHO*DELTA
      GO TO 10
50    WRITE(IOUTD,9997) KOUNT,FPSI
9997  FORMAT(//"FINAL VALUES- NO. OF FUNCT CALLS=",I5,
     +  7H FUNCT=,D20.10)
      RETURN
      END
C
C-------------------------------------------------------------
C SUBROUTINE:  EXPLR
C THIS IS A LOCAL EXPLORATION SUBROUTINE USED BY H-AND-J
C-------------------------------------------------------------
C
      SUBROUTINE EXPLR(PHI,FPHI,N,DELTA)
      DOUBLE PRECISION PHI(36),FPHI,DELTA,SAVE,FNEW,SAVEI,SAVEJ
      INTEGER LIST(36)
      COMMON/STEER/PRINT,TWOPT,FIXCOF
      LOGICAL PRINT,TWOPT,FIXCOF
C
C RANDOMIZE THE ORDER OF THE COORDINATES
C
      CALL SHUFF(N,LIST)
      ICOUNT = 1
10    IF(ICOUNT.GT.N) RETURN
      I = LIST(ICOUNT)
C
C IF ONLY THREE COEFFICIENTS WERE READ IN PER STAGE
C THEN DO NOT CHANGE THE SECOND NUMERATOR COEFFICIENT
C
      KSAVE = I-(I/4)*4
      IF(FIXCOF.AND.(KSAVE.EQ.2)) GO TO 100
      SAVE = PHI(I)
      PHI(I) = PHI(I)+DELTA
      CALL FUNCT(N,PHI,FNEW)
      IF(FNEW.GE.FPHI) GO TO 20
      FPHI = FNEW
      GO TO 100
20    PHI(I) = PHI(I)-2.D0*DELTA
      CALL FUNCT(N,PHI,FNEW)
      IF(FNEW.GE.FPHI) GO TO 30
      FPHI = FNEW
      GO TO 100
30    PHI(I) = SAVE
      IF(.NOT.TWOPT) GO TO 100
C
C BEGINNING OF 2-OPT
C
      JCOUNT = ICOUNT+1
40    IF(JCOUNT.GT.N) GO TO 100
      J = LIST(JCOUNT)
C
C SIMILARLY FOR TWOPT, IF THREE COEFFICIENTS PER STAGE WERE
```

```
C READ IN, DO NOT CHANGE THE SECOND NUMERATOR COEFFICIENT
C
      KSAVE = J-(J/4)*4
      IF(FIXCOF.AND.(KSAVE.EQ.2))  GO TO 90
      SAVEI = PHI(I)
      SAVEJ = PHI(J)
      PHI(I)  = PHI(I)  + DELTA
      PHI(J)  = PHI(J)  + DELTA
      CALL FUNCT(N,PHI,FNEW)
      IF(FNEW.GE.FPHI)  GO TO 50
      FPHI = FNEW
      GO TO 90
50    PHI(I)  = PHI(I)-2.D0*DELTA
      CALL FUNCT(N,PHI,FNEW)
      IF(FNEW.GE.FPHI)  GO TO 60
      FPHI = FNEW
      GO TO 90
60    PHI(J)  = PHI(J)-2.D0*DELTA
      CALL FUNCT(N,PHI,FNEW)
      IF(FNEW.GE.FPHI)  GO TO 70
      FPHI = FNEW
      GO TO 90
70    PHI(I)  = PHI(I)+2.D0*DELTA
      CALL FUNCT(N,PHI,FNEW)
      IF(FNEW.GE.FPHI)  GO TO 80
      FPHI = FNEW
      GO TO 90
80    PHI(I)  = SAVEI
      PHI(J)  = SAVEJ
90    JCOUNT  = JCOUNT+1
      GO TO 40
C
C END OF 2-OPT
C
100   ICOUNT  = ICOUNT+1
      GO TO 10
      END
C
C------------------------------------------------------------
C SUBROUTINE:   SET
C THIS SETS VECTOR A EQUAL TO VECTOR B
C------------------------------------------------------------
C
      SUBROUTINE SET(A,B,N)
      DOUBLE PRECISION A(36),B(36)
      DO 10 I=1,N
        A(I)  = B(I)
10    CONTINUE
      RETURN
      END
C
C------------------------------------------------------------
C SUBROUTINE:   SHUFF
C THIS RANDOMLY ORDERS LIST
C------------------------------------------------------------
```

```
C
      SUBROUTINE SHUFF(N,LIST)
      INTEGER LIST(36)
      DO 10 I=1,N
        LIST(I) = I
10    CONTINUE
      DO 20 LL=1,N
        L = N-LL+1
        J = INT(FLOAT(L)*UNI(0))+1
        K = LIST(L)
        LIST(L) = LIST(J)
        LIST(J) = K
20    CONTINUE
      RETURN
      END
C
C-----------------------------------------------------------------
C FUNCTION:  UNI
C AUTHOR:    ALAN M. GROSS
C            BELL LABORATORIES, MURRAY HILL, NEW JERSEY
C            07974
C PORTABLE RANDOM NUMBER GENERATOR
C-----------------------------------------------------------------
C
      FUNCTION UNI(K)
      INTEGER IBYTE(4)
      DATA ICSEED/0/,ITSEED/0/,IFCN/1/
C
      UNI=R1UNF(ICSEED,ITSEED,IBYTE,IFCN)
      RETURN
      END
C
      FUNCTION R1UNF(ICSEED,ITSEED,IBYTE,IFCN)
C
      LOGICAL FIRST
      INTEGER CSEED(6),TSEED(32),XOR(29),IBYTE(4),ISCR(5)
      DATA XOR(1)/1/,XOR(2)/2/,XOR(3)/3/,XOR(4)/3/,XOR(5)/2/,
     1 XOR(6)/1/,XOR(7)/4/,XOR(8)/5/,XOR(9)/6/,XOR(10)/7/,
     2 XOR(11)/5/,XOR(12)/4/,XOR(13)/7/,XOR(14)/6/,XOR(15)/1/,
     3 XOR(16)/6/,XOR(17)/7/,XOR(18)/4/,XOR(19)/5/,XOR(20)/2/,
     4 XOR(21)/3/,XOR(22)/7/,XOR(23)/6/,XOR(24)/5/,XOR(25)/4/,
     5 XOR(26)/3/,XOR(27)/2/,XOR(28)/1/,XOR(29)/0/
      DATA FIRST/.TRUE./,JCSEED/12345/,JTSEED/1073/
C
      DATA CSEED/0,0,0,0,0,0/
      DATA TSEED/0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
     1           0,0,0,0,0,0,0,0,0,0/
      R1UNF=0.0
      IF((.NOT.FIRST).AND.(IFCN.GT.0)) GO TO 50
      IF(IFCN.GT.0) GO TO 10
C
      JCSEED=IABS(ICSEED)
      JTSEED=IABS(ITSEED)
10    FIRST=.FALSE.
C
```

```
      CSEED(1) = JCSEED
      DO 20 I=1,5
      CSEED(I+1)=CSEED(I)/64
20    CSEED(I)=CSEED(I)-CSEED(I+1)*64
      CSEED(6)=MOD(CSEED(6),4)
C
      IF((JCSEED.NE.0).AND.(MOD(CSEED(1),2).EQ.0))
     +    CSEED(1)=CSEED(1)+1
      TSEED(1)=JTSEED
      DO 30 I=1,11
      TSEED(I+1)=TSEED(I)/2
30    TSEED(I)=TSEED(I)-TSEED(I+1)*2
C
      DO 40 I=12,32
40    TSEED(I)=0
      IF(JTSEED.NE.0) TSEED(1)=1
      IF(IFCN.EQ.0) RETURN
50    CONTINUE
      DO 60 I=1,17
60    TSEED(I)=IABS(TSEED(I)-TSEED(I+15))
      DO 70 I=18,32
70    TSEED(I)=IABS(TSEED(I)-TSEED(I-17))
C
      CSEED(6)=13*CSEED(6)+55*CSEED(5)*16*CSEED(4)
      CSEED(5)=13*CSEED(5)+55*CSEED(4)*16*CSEED(3)
      CSEED(4)=13*CSEED(4)+55*CSEED(3)*16*CSEED(2)
      CSEED(3)=13*CSEED(3)+55*CSEED(2)*16*CSEED(1)
      CSEED(2)=13*CSEED(2)+55*CSEED(1)
      CSEED(1)=13*CSEED(1)
      K=-5
      ICARRY=0
      DO 80 I=1,5
      K=K+6
      CSEED(I)=CSEED(I)+ICARRY
      ICARRY=CSEED(I)/64
      CSEED(I)=CSEED(I)-64*ICARRY
      I2=CSEED(I)/8
      I1=CSEED(I)-8*I2
      J1=4*TSEED(K+2)+TSEED(K+1)+TSEED(K+1)+TSEED(K)
      J2=4*TSEED(K+5)+TSEED(K+4)+TSEED(K+4)+TSEED(K+3)
      IT1=28
      IF(I1.GT.J1) IT1=(I1*I1-I1)/2+J1
      IF(I1.LT.J1) IT1=(J1*J1-J1)/2+I1
      IT2=28
      IF(I2.GT.J2) IT2=(I2*I2-I2)/2+J2
      IF(I2.LT.J2) IT2=(J2*J2-J2)/2+I2
      ISCR(I)=8*XOR(IT2+1)+XOR(IT1+1)
80    R1UNF=(R1UNF+FLOAT(ISCR(I)))/64.0
      CSEED(6)=MOD(CSEED(6)+ICARRY,4)
      J1=TSEED(31)+TSEED(32)+TSEED(32)
      IT1=CSEED(6)-J1
      IF(IT1.LT.0) IT1=0-IT1
      IF((IT1.EQ.1).AND.(CSEED(6)+J1.EQ.3)) IT1=3
      R1UNF=(R1UNF+FLOAT(IT1))/4.0
      IF(IFCN.EQ.1) RETURN
```

```
      IBYTE(4)=ISCR(1)+MOD(ISCR(2),4)*64
      IBYTE(3)=ISCR(2)/4+MOD(ISCR(3),16)*16
      IBYTE(2)=ISCR(3)/16+ISCR(4)*4
      IBYTE(1)=ISCR(5)+IT1*64
      RETURN
      END
C
      BLOCK DATA
      COMMON/ABC/X,NSET,NTWOPT,NBIT1,NBIT2,NSECT,N
      COMMON/RAW/W,Y,WEIGHT,M,KOUNT
      COMMON/STEER/PRINT,TWOPT,FIXCOF
      COMMON/MACH/IND,IOUTD
      DOUBLE PRECISION W(80),Y(80),WEIGHT(80),X(36)
      END
```

FWIIR Input

```
3,100,1,6,5
0.0         0.0         .01
.36         0.0         .01
.3658       0.0         .01
.383333     0.0         .01
.411111     1.0         .03
.412        1.0         .03
.4129       1.0         .03
.414        1.0         .03
.4157       1.0         .03
.4185       1.0         .03
.423        1.0         .03
.4275       1.0         .03
.433        1.0         .03
.4386       1.0         .03
.4443       1.0         .03
.45         1.0         .03
.4545       1.0         .03
.459        1.0         .03
.461916     1.0         .03
.4636       1.0         .03
.464833     1.0         .03
.465749     1.0         .03
.466666     1.0         .03
.494444     0.0         .01
.51         0.0         .01
.51448      0.0         .01
.52         0.0         .01
1.0         0.0         .01
4
 0.4512591755
-0.2855823838
 0.9116860616
-1.0986945504
-0.4457342317
 0.9130786730
-0.0099665157
-0.1969016650
 0.9695353354
-0.7304169706
-0.5515388641
 0.9705899009
```

FWIIR Output

```
***** INPUT DATA *****
NUMBER OF COEFFICIENTS PER STAGE IS  3
RANDOM INITIALIZATION IS 100
TWOPT IS  1
FINAL ( AND INITIAL ROUNDING ) PRECISION IS  6 BITS
INITIAL SEARCH DELTA IS  5 BITS
```

GRID SPECIFICATIONS

POINT NO.	FREQUENCY	DESIRED Y	TOLERANCE
1	.0000000000D+00	.0000000000D+00	1.0000000000D-02
2	.3600000000D+00	.0000000000D+00	1.0000000000D-02
3	.3658000000D+00	.0000000000D+00	1.0000000000D-02
4	.3833330000D+00	.0000000000D+00	1.0000000000D-02
5	.4111110000D+00	.1000000000D+01	.3000000000D-01
6	.4120000000D+00	.1000000000D+01	.3000000000D-01
7	.4129000000D+00	.1000000000D+01	.3000000000D-01
8	.4140000000D+00	.1000000000D+01	.3000000000D-01
9	.4157000000D+00	.1000000000D+01	.3000000000D-01
10	.4185000000D+00	.1000000000D+01	.3000000000D-01
11	.4230000000D+00	.1000000000D+01	.3000000000D-01
12	.4275000000D+00	.1000000000D+01	.3000000000D-01
13	.4330000000D+00	.1000000000D+01	.3000000000D-01
14	.4386000000D+00	.1000000000D+01	.3000000000D-01
15	.4443000000D+00	.1000000000D+01	.3000000000D-01
16	.4500000000D+00	.1000000000D+01	.3000000000D-01
17	.4545000000D+00	.1000000000D+01	.3000000000D-01
18	.4590000000D+00	.1000000000D+01	.3000000000D-01
19	.4619160000D+00	.1000000000D+01	.3000000000D-01
20	.4636000000D+00	.1000000000D+01	.3000000000D-01
21	.4648330000D+00	.1000000000D+01	.3000000000D-01
22	.4657490000D+00	.1000000000D+01	.3000000000D-01
23	.4666660000D+00	.1000000000D+01	.3000000000D-01
24	.4944440000D+00	.0000000000D+00	1.0000000000D-02
25	.5100000000D+00	.0000000000D+00	1.0000000000D-02
26	.5144800000D+00	.0000000000D+00	1.0000000000D-02
27	.5200000000D+00	.0000000000D+00	1.0000000000D-02
28	.1000000000D+01	.0000000000D+00	1.0000000000D-02

```
INITIAL HIGH PRECISION COEFFICIENTS FOR  4 SECTIONS ARE
SECTION  1
      .4512591755D+00      .1000000000D+01
     -.2855823838D+00      .9116860616D+00
SECTION  2
     -.1099694550D+01      .1000000000D+01
     -.4457342317D+00      .9130786730D+00
SECTION  3
     -.9966515700D-02      .1000000000D+01
     -.1969016650D+00      .9695353354D+00
SECTION  4
     -.7304169706D+00      .1000000000D+01
     -.5515388641D+00      .9705899009D+00


THE INITIAL COEFFICIENTS, ROUNDED TO   6 BITS, ARE
SECTION  1
      .4531250000D+00      .1000000000D+01
     -.2812500000D+00      .9062500000D+00
SECTION  2
     -.1093750000D+01      .1000000000D+01
     -.4531250000D+00      .9062500000D+00
SECTION  3
     -.1562500000D-01      .1000000000D+01
     -.2031250000D+00      .9687500000D+00
SECTION  4
     -.7343750000D+00      .1000000000D+01
     -.5468750000D+00      .9687500000D+00

NEXT FOLLOWS A REPORT FROM THE SEARCH ALGORITHM
NO. CALLS  PAT. SIZE           DELTA       OLD VALUE        NEW VALUE
      248          1   .31250000D-01   .25259811D+01   .22551520D+01
      560          1   .31250000D-01   .22551520D+01   .21501752D+01
      806          2   .31250000D-01   .21501752D+01   .17458384D+01
     1378          1   .31250000D-01   .17458384D+01   .16952136D+01
     2213          1   .15625000D-01   .16952136D+01   .74287695D+00
     2721          1   .15625000D-01   .74287695D+00   .73309429D+00
     2998          2   .15625000D-01   .73309429D+00   .71935858D+00
```

```
FINAL VALUES- NO. OF FUNCT CALLS=  3537 FUNCT=      .7193585794D+00

THE FINAL COEFFICIENTS, HAVING  6 BITS, ARE
SECTION  1
       .4375000000D+00      .1000000000D+01
      -.2812500000D+00      .9062500000D+00
SECTION  2
      -.1093750000D+01      .1000000000D+01
      -.4531250000D+00      .9062500000D+00
SECTION  3
      -.1562500000D-01      .1000000000D+01
      -.1875000000D+00      .9687500000D+00
SECTION  4
      -.7343750000D+00      .1000000000D+01
      -.5625000000D+00      .9687500000D+00
```

***** FINAL RESULTS *****

THE CONSTANT A IS .7493357724D-02

FUNCTION VALUE ON COARSE GRID IS .7193585794D+00

FREQUENCY	DESIRED Y	ACTUAL Y	ERROR
.00000000D+00	.00000000D+00	.70283624D-02	.70283624D+00
.36000000D+00	.00000000D+00	.64137195D-02	.64137195D+00
.36580000D+00	.00000000D+00	.67483317D-02	.67483317D+00
.38333300D+00	.00000000D+00	.52589490D-02	.52589490D+00
.41111100D+00	.10000000D+01	.10215808D+01	.71935858D+00
.41200000D+00	.10000000D+01	.10204763D+01	.68254419D+00
.41290000D+00	.10000000D+01	.10148222D+01	.49407251D+00
.41400000D+00	.10000000D+01	.10066453D+01	.22150991D+00
.41570000D+00	.10000000D+01	.99765910D+00	.78029934D-01
.41850000D+00	.10000000D+01	.99730469D+00	.89843802D-01
.42300000D+00	.10000000D+01	.10136004D+01	.45334536D+00
.42750000D+00	.10000000D+01	.10161582D+01	.53860583D+00
.43300000D+00	.10000000D+01	.99671587D+00	.10947108D+00
.43860000D+00	.10000000D+01	.98587547D+00	.47081780D+00
.44430000D+00	.10000000D+01	.10004271D+01	.14235640D-01
.45000000D+00	.10000000D+01	.10154313D+01	.51437769D+00

.45450000D+00	.10000000D+01	.10021237D+01	.70791248D-01
.45900000D+00	.10000000D+01	.97841924D+00	.71935858D+00
.46191600D+00	.10000000D+01	.97893618D+00	.70546056D+00
.46360000D+00	.10000000D+01	.98927917D+00	.35736098D+00
.46433300D+00	.10000000D+01	.99955201D+00	.14933148D-01
.46574900D+00	.10000000D+01	.10052380D+01	.17460152D+00
.46666600D+00	.10000000D+01	.10045265D+01	.15088407D+00
.49444400D+00	.00000000D+00	.53608917D-02	.53608917D+00
.51000000D+00	.00000000D+00	.70073328D-02	.70073328D+00
.51448000D+00	.00000000D+00	.71485086D-02	.71485086D+00
.52000000D+00	.00000000D+00	.67435744D-02	.67435744D+00
.10000000D+01	.00000000D+00	.70871289D-02	.70871289D+00

FINAL RESULTS ON FINE GRID- 10 POINTS PER GRID POINT

FREQUENCY	DESIRED Y	ACTUAL Y	ERROR
***** BAND FROM GRID POINT 1 TO 2 *****			
.00000000D+00	.00000000D+00	.70355890D-02	.70355890D+00
.36000000D-01	.00000000D+00	.70067577D-02	.70067577D+00
.72000000D-01	.00000000D+00	.69162904D-02	.69162904D+00
.10800000D+00	.00000000D+00	.67511103D-02	.67511103D+00
.14400000D+00	.00000000D+00	.64849851D-02	.64849851D+00
.18000000D+00	.00000000D+00	.60693616D-02	.60693616D+00
.21600000D+00	.00000000D+00	.54133609D-02	.54133609D+00
.25200000D+00	.00000000D+00	.43383097D-02	.43383097D+00
.28800000D+00	.00000000D+00	.24698003D-02	.24698003D+00
.32400000D+00	.00000000D+00	.98742480D-03	.98742480D-01
***** BAND FROM GRID POINT 2 TO 3 *****			
.36000000D+00	.00000000D+00	.64203140D-02	.64203140D+00
.36058000D+00	.00000000D+00	.64798329D-02	.64798329D+00
.36116000D+00	.00000000D+00	.65348298D-02	.65348298D+00
.36174000D+00	.00000000D+00	.65848927D-02	.65848927D+00
.36232000D+00	.00000000D+00	.66295783D-02	.66295783D+00
.36290000D+00	.00000000D+00	.66684091D-02	.66684091D+00
.36348000D+00	.00000000D+00	.67008707D-02	.67008707D+00
.36406000D+00	.00000000D+00	.67264089D-02	.67264089D+00
.36464000D+00	.00000000D+00	.67444262D-02	.67444262D+00

.36522000D+00	.00000000D+00	.67542783D-02	.67542783D+00
***** BAND FROM GRID POINT 3 TO 4 *****			
.36580000D+00	.00000000D+00	.67552702D-02	.67552702D+00
.36755330D+00	.00000000D+00	.66964453D-02	.66964453D+00
.36930660D+00	.00000000D+00	.65250872D-02	.65250872D+00
.37105990D+00	.00000000D+00	.62103616D-02	.62103616D+00
.37281320D+00	.00000000D+00	.57130145D-02	.57130145D+00
.37456650D+00	.00000000D+00	.49827859D-02	.49827859D+00
.37631980D+00	.00000000D+00	.39549045D-02	.39549045D+00
.37807310D+00	.00000000D+00	.25452881D-02	.25452881D+00
.37982640D+00	.00000000D+00	.64388720D-03	.64388720D-01
.38157970D+00	.00000000D+00	.18946617D-02	.18946617D+00
.38333300D+00	.00000000D+00	.52643562D-02	.52643562D+00
***** BAND FROM GRID POINT 5 TO 6 *****			
.41111100D+00	.10000000D+01	.10226311D+01	.75437138D+00
.41119990D+00	.10000000D+01	.10228557D+01	.76185759D+00
.41128880D+00	.10000000D+01	.10229910D+01	.76636673D+00
.41137770D+00	.10000000D+01	.10230429D+01	.76809520D+00
.41146660D+00	.10000000D+01	.10230170D+01	.76723329D+00
.41155550D+00	.10000000D+01	.10229190D+01	.76396511D+00
.41164440D+00	.10000000D+01	.10227540D+01	.75846829D+00
.41173330D+00	.10000000D+01	.10225274D+01	.75091396D+00
.41182220D+00	.10000000D+01	.10222440D+01	.74146672D+00
.41191110D+00	.10000000D+01	.10219085D+01	.73028463D+00
***** BAND FROM GRID POINT 6 TO 7 *****			
.41200000D+00	.10000000D+01	.10215256D+01	.71751914D+00
.41209000D+00	.10000000D+01	.10210939D+01	.70313100D+00
.41218000D+00	.10000000D+01	.10206224D+01	.68741224D+00
.41227000D+00	.10000000D+01	.10201150D+01	.67049987D+00
.41236000D+00	.10000000D+01	.10195757D+01	.65252422D+00
.41245000D+00	.10000000D+01	.10190083D+01	.63360933D+00
.41254000D+00	.10000000D+01	.10184162D+01	.61387285D+00
.41263000D+00	.10000000D+01	.10178028D+01	.59342635D+00
.41272000D+00	.10000000D+01	.10171713D+01	.57237539D+00
.41281000D+00	.10000000D+01	.10165246D+01	.55081977D+00
***** BAND FROM GRID POINT 7 TO 8 *****			
.41290000D+00	.10000000D+01	.10158656D+01	.52885367D+00

.41301000D+00	.10000000D+01	.10150473D+01	.50157786D+00
.41312000D+00	.10000000D+01	.10142192D+01	.47397264D+00
.41323000D+00	.10000000D+01	.10133854D+01	.44617840D+00
.41334000D+00	.10000000D+01	.10125497D+01	.41832484D+00
.41345000D+00	.10000000D+01	.10117159D+01	.39053156D+00
.41356000D+00	.10000000D+01	.10108873D+01	.36290857D+00
.41367000D+00	.10000000D+01	.10100667D+01	.33555676D+00
.41378000D+00	.10000000D+01	.10092571D+01	.30856840D+00
.41389000D+00	.10000000D+01	.10084608D+01	.28202761D+00
***** BAND FROM GRID POINT 8 TO 9 *****			
.41400000D+00	.10000000D+01	.10076803D+01	.25601082D+00
.41417000D+00	.10000000D+01	.10065098D+01	.21699242D+00
.41434000D+00	.10000000D+01	.10053884D+01	.17961280D+00
.41451000D+00	.10000000D+01	.10043218D+01	.14406042D+00
.41468000D+00	.10000000D+01	.10033147D+01	.11049159D+00
.41485000D+00	.10000000D+01	.10023710D+01	.79033587D-01
.41502000D+00	.10000000D+01	.10014936D+01	.49787471D-01
.41519000D+00	.10000000D+01	.10006849D+01	.22830602D-01
.41536000D+00	.10000000D+01	.99994657D+00	.17809056D-02
.41553000D+00	.10000000D+01	.99927970D+00	.24010060D-01
***** BAND FROM GRID POINT 9 TO 10 *****			
.41570000D+00	.10000000D+01	.99868489D+00	.43837008D-01
.41598000D+00	.10000000D+01	.99786249D+00	.71250443D-01
.41626000D+00	.10000000D+01	.99723405D+00	.92198193D-01
.41654000D+00	.10000000D+01	.99679534D+00	.10682199D+00
.41682000D+00	.10000000D+01	.99654003D+00	.11533238D+00
.41710000D+00	.10000000D+01	.99646012D+00	.11799615D+00
.41738000D+00	.10000000D+01	.99654623D+00	.11512559D+00
.41766000D+00	.10000000D+01	.99678792D+00	.10706936D+00
.41794000D+00	.10000000D+01	.99717396D+00	.94204726D-01
.41822000D+00	.10000000D+01	.99769207D+00	.76930848D-01
***** BAND FROM GRID POINT 10 TO 11 *****			
.41850000D+00	.10000000D+01	.99833011D+00	.55663023D-01
.41895000D+00	.10000000D+01	.99957424D+00	.14192097D-01
.41940000D+00	.10000000D+01	.10010407D+01	.34690603D-01
.41985000D+00	.10000000D+01	.10026754D+01	.89178770D-01
.42030000D+00	.10000000D+01	.10044248D+01	.14749383D+00

.42075000D+00	.10000000D+01	.10042374D+01	.20791349D+00
.42120000D+00	.10000000D+01	.10030639D+01	.26879640D+00
.42165000D+00	.10000000D+01	.10098581D+01	.32860353D+00
.42210000D+00	.10000000D+01	.10115775D+01	.38591672D+00
.42255000D+00	.10000000D+01	.10131836D+01	.43945441D+00
♦♦♦♦♦ BAND FROM GRID POINT 11 TO 12 ♦♦♦♦♦			
.42300000D+00	.10000000D+01	.10146425D+01	.48808464D+00
.42345000D+00	.10000000D+01	.10159251D+01	.53083510D+00
.42390000D+00	.10000000D+01	.10170070D+01	.56690020D+00
.42435000D+00	.10000000D+01	.10178694D+01	.59564509D+00
.42480000D+00	.10000000D+01	.10184982D+01	.61660647D+00
.42525000D+00	.10000000D+01	.10188847D+01	.62949036D+00
.42570000D+00	.10000000D+01	.10190250D+01	.63416685D+00
.42615000D+00	.10000000D+01	.10189199D+01	.63066215D+00
.42660000D+00	.10000000D+01	.10185744D+01	.61914821D+00
.42705000D+00	.10000000D+01	.10179979D+01	.59993027D+00
♦♦♦♦♦ BAND FROM GRID POINT 12 TO 13 ♦♦♦♦♦			
.42750000D+00	.10000000D+01	.10172030D+01	.57343278D+00
.42805000D+00	.10000000D+01	.10159532D+01	.53193936D+00
.42860000D+00	.10000000D+01	.10144452D+01	.48150636D+00
.42915000D+00	.10000000D+01	.10127028D+01	.42342625D+00
.42970000D+00	.10000000D+01	.10107730D+01	.35910085D+00
.43025000D+00	.10000000D+01	.10087000D+01	.29000025D+00
.43080000D+00	.10000000D+01	.10065288D+01	.21762508D+00
.43135000D+00	.10000000D+01	.10043042D+01	.14347267D+00
.43190000D+00	.10000000D+01	.10020702D+01	.69007614D-01
.43245000D+00	.10000000D+01	.99986911D+00	.43630795D-02
♦♦♦♦♦ BAND FROM GRID POINT 13 TO 14 ♦♦♦♦♦			
.43300000D+00	.10000000D+01	.99774069D+00	.75310484D-01
.43356000D+00	.10000000D+01	.99566653D+00	.14378247D+00
.43412000D+00	.10000000D+01	.99378179D+00	.20727376D+00
.43468000D+00	.10000000D+01	.99205838D+00	.26472066D+00
.43524000D+00	.10000000D+01	.99054424D+00	.31519205D+00
.43580000D+00	.10000000D+01	.98926317D+00	.35789439D+00
.43636000D+00	.10000000D+01	.98823476D+00	.39217481D+00
.43692000D+00	.10000000D+01	.98747431D+00	.41752304D+00
.43748000D+00	.10000000D+01	.98699283D+00	.43357240D+00

```
 .43804000D+00   .10000000D+01   .93679699D+00   .44010034D+00
***** BAND FROM GRID POINT   14 TO   15 *****
 .43860000D+00   .10000000D+01   .98688914D+00   .43702874D+00
 .43917000D+00   .10000000D+01   .98727660D+00   .42411346D+00
 .43974000D+00   .10000000D+01   .98795361D+00   .40154633D+00
 .44031000D+00   .10000000D+01   .98890878D+00   .36970726D+00
 .44088000D+00   .10000000D+01   .99012605D+00   .32913180D+00
 .44145000D+00   .10000000D+01   .99158468D+00   .28051072D+00
 .44202000D+00   .10000000D+01   .99325934D+00   .22468873D+00
 .44259000D+00   .10000000D+01   .99512014D+00   .16266198D+00
 .44316000D+00   .10000000D+01   .99713279D+00   .95573681D-01
 .44373000D+00   .10000000D+01   .99925878D+00   .24707292D-01
***** BAND FROM GRID POINT   15 TO   16 *****
 .44430000D+00   .10000000D+01   .10014557D+01   .48523432D-01
 .44487000D+00   .10000000D+01   .10036776D+01   .12258826D+00
 .44544000D+00   .10000000D+01   .10058758D+01   .19585877D+00
 .44601000D+00   .10000000D+01   .10079990D+01   .26663178D+00
 .44658000D+00   .10000000D+01   .10099947D+01   .33315816D+00
 .44715000D+00   .10000000D+01   .10118103D+01   .39367739D+00
 .44772000D+00   .10000000D+01   .10133937D+01   .44645761D+00
 .44829000D+00   .10000000D+01   .10146952D+01   .48984070D+00
 .44886000D+00   .10000000D+01   .10156687D+01   .52229133D+00
 .44943000D+00   .10000000D+01   .10162735D+01   .54244881D+00
***** BAND FROM GRID POINT   16 TO   17 *****
 .45000000D+00   .10000000D+01   .10164754D+01   .54917973D+00
 .45045000D+00   .10000000D+01   .10163331D+01   .54443717D+00
 .45090000D+00   .10000000D+01   .10159152D+01   .53050824D+00
 .45135000D+00   .10000000D+01   .10152176D+01   .50725413D+00
 .45180000D+00   .10000000D+01   .10142406D+01   .47468723D+00
 .45225000D+00   .10000000D+01   .10129894D+01   .43298007D+00
 .45270000D+00   .10000000D+01   .10114741D+01   .38247162D+00
 .45315000D+00   .10000000D+01   .10097101D+01   .32367066D+00
 .45360000D+00   .10000000D+01   .10077177D+01   .25725630D+00
 .45405000D+00   .10000000D+01   .10055223D+01   .18407541D+00
***** BAND FROM GRID POINT   17 TO   18 *****
 .45450000D+00   .10000000D+01   .10031541D+01   .10513719D+00
 .45495000D+00   .10000000D+01   .10006481D+01   .21604979D-01
```

.45540000D+00	.10000000D+01	.99804357D+00	.65214441D-01
.45585000D+00	.10000000D+01	.99538349D+00	.15388373D+00
.45630000D+00	.10000000D+01	.99271451D+00	.24284983D+00
.45675000D+00	.10000000D+01	.99008619D+00	.33046047D+00
.45720000D+00	.10000000D+01	.98755053D+00	.41498223D+00
.45765000D+00	.10000000D+01	.98516139D+00	.49462017D+00
.45810000D+00	.10000000D+01	.98297382D+00	.56753922D+00
.45855000D+00	.10000000D+01	.98104336D+00	.63188788D+00
***** BAND FROM GRID POINT 18 TO 19 *****			
.45900000D+00	.10000000D+01	.97942525D+00	.68582506D+00
.45929160D+00	.10000000D+01	.97856901D+00	.71436630D+00
.45958320D+00	.10000000D+01	.97788085D+00	.73730516D+00
.45987480D+00	.10000000D+01	.97737438D+00	.75418722D+00
.46016640D+00	.10000000D+01	.97706251D+00	.76458306D+00
.46045800D+00	.10000000D+01	.97695714D+00	.76809520D+00
.46074960D+00	.10000000D+01	.97706901D+00	.76436644D+00
.46104120D+00	.10000000D+01	.97740731D+00	.75308964D+00
.46133280D+00	.10000000D+01	.97797942D+00	.73401943D+00
.46162440D+00	.10000000D+01	.97879041D+00	.70698620D+00
***** BAND FROM GRID POINT 19 TO 20 *****			
.46191600D+00	.10000000D+01	.97984262D+00	.67191276D+00
.46208440D+00	.10000000D+01	.98055989D+00	.64800377D+00
.46225280D+00	.10000000D+01	.98135653D+00	.62144903D+00
.46242120D+00	.10000000D+01	.98223124D+00	.59229187D+00
.46258960D+00	.10000000D+01	.98318217D+00	.56059431D+00
.46275800D+00	.10000000D+01	.98420683D+00	.52643913D+00
.46292640D+00	.10000000D+01	.98530204D+00	.48993216D+00
.46309480D+00	.10000000D+01	.98646386D+00	.45120477D+00
.46326320D+00	.10000000D+01	.98768750D+00	.41041656D+00
.46343160D+00	.10000000D+01	.98896725D+00	.36775835D+00
***** BAND FROM GRID POINT 20 TO 21 *****			
.46360000D+00	.10000000D+01	.99029634D+00	.32345526D+00
.46372330D+00	.10000000D+01	.99129627D+00	.29012439D+00
.46384660D+00	.10000000D+01	.99231493D+00	.25616890D+00
.46396990D+00	.10000000D+01	.99334844D+00	.22171855D+00
.46409320D+00	.10000000D+01	.99439250D+00	.18691667D+00
.46421650D+00	.10000000D+01	.99544237D+00	.15192097D+00

.46433980D+00	.10000000D+01	.99649237D+00	.11690443D+00
.46446310D+00	.10000000D+01	.99755531D+00	.82056192D-01
.46458640D+00	.10000000D+01	.99857252D+00	.47582575D-01
.46470970D+00	.10000000D+01	.99958876D+00	.13707874D-01
♦♦♦♦♦ BAND FROM GRID POINT 21 TO 22 ♦♦♦♦♦			
.46483300D+00	.10000000D+01	.10005797D+01	.19324653D-01
.46492460D+00	.10000000D+01	.10012949D+01	.43164441D-01
.46501620D+00	.10000000D+01	.10019884D+01	.66279399D-01
.46510780D+00	.10000000D+01	.10026564D+01	.88547773D-01
.46519940D+00	.10000000D+01	.10032952D+01	.10984114D+00
.46529100D+00	.10000000D+01	.10039007D+01	.13002423D+00
.46538260D+00	.10000000D+01	.10044686D+01	.14895476D+00
.46547420D+00	.10000000D+01	.10049945D+01	.16648333D+00
.46556580D+00	.10000000D+01	.10054736D+01	.18245331D+00
.46565740D+00	.10000000D+01	.10059010D+01	.19670071D+00
♦♦♦♦♦ BAND FROM GRID POINT 22 TO 23 ♦♦♦♦♦			
.46574900D+00	.10000000D+01	.10062716D+01	.20905420D+00
.46584070D+00	.10000000D+01	.10065804D+01	.21934505D+00
.46593240D+00	.10000000D+01	.10068212D+01	.22737192D+00
.46602410D+00	.10000000D+01	.10069882D+01	.23294069D+00
.46611580D+00	.10000000D+01	.10070755D+01	.23584994D+00
.46620750D+00	.10000000D+01	.10070767D+01	.23589117D+00
.46629920D+00	.10000000D+01	.10069855D+01	.23284895D+00
.46639090D+00	.10000000D+01	.10067950D+01	.22650122D+00
.46648260D+00	.10000000D+01	.10064986D+01	.21661961D+00
.46657430D+00	.10000000D+01	.10060891D+01	.20296987D+00
.46666600D+00	.10000000D+01	.10055594D+01	.18531236D+00
♦♦♦♦♦ BAND FROM GRID POINT 24 TO 25 ♦♦♦♦♦			
.49444400D+00	.00000000D+00	.53664037D-02	.53664037D+00
.49599960D+00	.00000000D+00	.23290166D-02	.23290166D+00
.49755520D+00	.00000000D+00	.57133102D-04	.57133102D-02
.49911080D+00	.00000000D+00	.19262911D-02	.19262911D+00
.50066640D+00	.00000000D+00	.33826997D-02	.33826997D+00
.50222200D+00	.00000000D+00	.45079748D-02	.45079748D+00
.50377760D+00	.00000000D+00	.53664973D-02	.53664973D+00
.50533320D+00	.00000000D+00	.60093984D-02	.60093984D+00
.50688880D+00	.00000000D+00	.64775366D-02	.64775366D+00

```
  .50844440D+00   .00000000D+00   .68037445D-02   .68037445D+00
***** BAND FROM GRID POINT    25 TO    26 *****
  .51000000D+00   .00000000D+00   .70145377D-02   .70145377D+00
  .51044800D+00   .00000000D+00   .70569898D-02   .70569898D+00
  .51089600D+00   .00000000D+00   .70921266D-02   .70921266D+00
  .51134400D+00   .00000000D+00   .71203702D-02   .71203702D+00
  .51179200D+00   .00000000D+00   .71421190D-02   .71421190D+00
  .51224000D+00   .00000000D+00   .71577488D-02   .71577488D+00
  .51268800D+00   .00000000D+00   .71676143D-02   .71676143D+00
  .51313600D+00   .00000000D+00   .71720504D-02   .71720504D+00
  .51358400D+00   .00000000D+00   .71713735D-02   .71713735D+00
  .51403200D+00   .00000000D+00   .71658821D-02   .71658821D+00
***** BAND FROM GRID POINT    26 TO    27 *****
  .51448000D+00   .00000000D+00   .71558587D-02   .71558587D+00
  .51503200D+00   .00000000D+00   .71376700D-02   .71376700D+00
  .51558400D+00   .00000000D+00   .71134736D-02   .71134736D+00
  .51613600D+00   .00000000D+00   .70837070D-02   .70837070D+00
  .51668800D+00   .00000000D+00   .70487786D-02   .70487786D+00
  .51724000D+00   .00000000D+00   .70090699D-02   .70090699D+00
  .51779200D+00   .00000000D+00   .69649372D-02   .69649372D+00
  .51834400D+00   .00000000D+00   .69167135D-02   .69167135D+00
  .51889600D+00   .00000000D+00   .68647099D-02   .68647099D+00
  .51944800D+00   .00000000D+00   .68092175D-02   .68092175D+00
***** BAND FROM GRID POINT    27 TO    28 *****
  .52000000D+00   .00000000D+00   .67505081D-02   .67505081D+00
  .56800000D+00   .00000000D+00   .22088925D-03   .22088925D-01
  .61600000D+00   .00000000D+00   .31906681D-02   .31906681D+00
  .66400000D+00   .00000000D+00   .48591220D-02   .48591220D+00
  .71200000D+00   .00000000D+00   .57711975D-02   .57711975D+00
  .76000000D+00   .00000000D+00   .63127353D-02   .63127353D+00
  .80800000D+00   .00000000D+00   .66508561D-02   .66508561D+00
  .85600000D+00   .00000000D+00   .68657264D-02   .68657264D+00
  .90400000D+00   .00000000D+00   .69986082D-02   .69986082D+00
  .95200000D+00   .00000000D+00   .70712714D-02   .70712714D+00
  .10000000D+01   .00000000D+00   .70944158D-02   .70944158D+00
```

```
THE VALUE OF A FROM THE FINE GRID IS      .7501062355D-02

THE FINAL VALUE OF THE ERROR ON THE FINE GRID IS      .7680951994D+00
```

Appendix E: A/D Converter Software Listings

Two methods of software support for the A/D converters were implemented. A third method was only partially implemented. Program ANDIG uses a polling loop to acquire data. It can call one of two subroutines, AD14B or AD15B, depending on which I/O channel the user selects. These two subroutines initiate the A/D conversion, load the data value into the A register, and then store it in a memory buffer location. This process repeats until the buffer is full. The subroutine WAIT1 is called to wait in a loop a specified amount of time until the initiation of the next A/D conversion. An alternate implementation of ANDIG uses the subroutines AD14 and AD15 (same entry points: DT14B, DT15B). These subroutines were used to measure the approximate conversion times of AD14B and AD15B.

Program ANDG2 uses a DMA channel to acquire data. It can call one of two subroutines, DMA14 or DMA15, depending on which I/O channel the user selects. These subroutines check for the availability of DCPC channel 1, initiate the data transfer, and wait in a loop until the transfer is complete. It is possible to execute other instructions rather than waiting in the loop.

The three dummy drivers listed are identical in function and differ only in the names of their respective entry points. The listings provide sufficient explanation of their purpose and implementation. They were included in the latest generation of RTE-III.

```
FTN4,L
C
C
      PROGRAM ANDIG
C
C**********************************************************************
C*                                                                   *
C*    THIS PROGRAM TESTS THE SUBROUTINES AD14B AND                   *
C*    AD15B.  THE ENTRY POINTS ARE DT14B AND DT15B,                  *
C*    RESPECTIVELY.                                                  *
C*                                                                   *
C**********************************************************************
C
      INTEGER DELAY,DATA,BUFL,DECIDE
      COMMON DELAY,DATA(8192),BUFL
C
C---  INITIALIZE VARIABLES
C
      WRITE(1,100)
      READ(1,*) DELAY
      WRITE(1,200)
      READ(1,*) BUFL
      WRITE(1,300)
      READ(1,*) LUNUM
      WRITE(1,400)
      READ(1,*) DECIDE
C
C---  GET THE DATA
C
      IF(DECIDE.EQ.14) CALL DT14B
      IF(DECIDE.EQ.15) CALL DT15B
C
C---  OUTPUT DATA
C
      WRITE(LUNUM,500) (I,DATA(I),I=1,BUFL)
C
100   FORMAT(/////,"CHOOSE A DELAY VALUE FROM 0 TO 32767. ")
C
200   FORMAT(//,"HOW MANY DATA POINTS?  (8192 MAX) ")
C
300   FORMAT(//,"WHAT IS THE LU# OF THE OUTPUT DEVICE? ")
C
400   FORMAT(//,"WHICH I/O CHANNEL, 14 OR 15?")
C
500   FORMAT(22(/),128(4(2I7),/))
C
      STOP
      END
```

```
ASMB,L
*
*
      NAM AD14B
*
*****************************************************
*                                                   *
*     THIS SUBROUTINE USES POLLING SOFTWARE TO      *
*     GET DATA FROM AN A/D CONVERTER.  THE I/O      *
*     SELECT CODE IS 14.  MAXIMUM NUMBER OF DATA    *
*     WORDS IS 8192.                                *
*                                                   *
*     VARIABLES:                                    *
*       DELAY - ADDS DELAY TIME BETWEEN THE ACQUI-  *
*               SITION OF EACH TWO DATA POINTS      *
*       DATA - DATA BUFFER                          *
*       BUFL - LENGTH OF THE DATA BUFFER            *
*                                                   *
*****************************************************
*
      EXT $LIBR,$LIBX,WAIT
      ENT DT14B
*
      COM DELAY,DATA(8192),BUFL
*
DT14B NOP
      JSB $LIBR       * TURN OFF INTERRUPTS AND MEMORY PROTECT
      NOP             *
      LDA DELAY       *
      SSA             * SKIP IF DELAY>0 OR =0
      CMA             * COMPLEMENT IF DELAY<0
      STA DELAY       *
      LDX BUFL        * SET COUNTER
      LDB STORE       * SET BUFFER POINTER
*
LOOP  STC 14B,C       * SEND ENCODE SIGNAL
      SFS 14B         * IS DEVICE BUSY?
      JMP *-1         * YES. TRY AGAIN.
      LIA 14B         * NO. READ DATA.
      STA STORE,I     * STORE DATA IN BUFFER
      INB             *
      STB STORE       * SET POINTER TO NEXT BUFFER LOCATION
      JSB WAIT        * DELAY PERIOD BEFORE GETTING NEW DATA
      DSX             * DECREMENT COUNT. SKIP AFTER LAST DATA PT.
      JMP LOOP        * GET NEXT DATA
*
      CLC 14B,C       * CLEAR I/O CONTROL BIT
      JSB $LIBX       * TURN ON INTERRUPTS AND MEMORY PROTECT
      DEF DT14B       * RETURN TO CALLING PROGRAM
*
STORE DEF DATA        * DEFINE DATA BUFFER ADDRESS
*
      END DT14B
```

```
ASMB,L
*
*
      NAM AD15B
*
*****************************************************
*                                                   *
*     THIS SUBROUTINE USES POLLING SOFTWARE TO      *
*     GET DATA FROM AN A/D CONVERTER.  THE I/O      *
*     SELECT CODE IS 15.  MAXIMUM NUMBER OF DATA    *
*     WORDS IS 8192.                                *
*                                                   *
*     VARIABLES:                                    *
*       DELAY - ADDS DELAY TIME BETWEEN THE ACQUI-  *
*               SITION OF EACH TWO DATA POINTS      *
*       DATA  - DATA BUFFER                         *
*       BUFL  - LENGTH OF THE DATA BUFFER           *
*                                                   *
*****************************************************
*
      EXT $LIBR,$LIBX,WAIT
      ENT DT15B
*
      COM DELAY,DATA(8192),BUFL
*
DT15B NOP
      JSB $LIBR       * TURN OFF INTERRUPTS AND MEMORY PROTECT
      NOP             *
      LDA DELAY       *
      SSA             * SKIP IF DELAY>0 OR =0
      CMA             * COMPLEMENT IF DELAY<0
      STA DELAY       *
      LDX BUFL        * SET COUNTER
      LDB STORE       * SET BUFFER POINTER
*
LOOP  STC 15B,C       * SEND ENCODE SIGNAL
      SFS 15B         * IS DEVICE BUSY?
      JMP *-1         * YES. TRY AGAIN.
      LIA 15B         * NO. READ DATA.
      STA STORE,I     * STORE DATA IN BUFFER
      INB             *
      STB STORE       * SET POINTER TO NEXT BUFFER LOCATION
      JSB WAIT        * DELAY PERIOD BEFORE GETTING NEW DATA
      DSX             * DECREMENT COUNT. SKIP AFTER LAST DATA PT.
      JMP LOOP        * GET NEXT DATA
*
      CLC 15B,C       * CLEAR I/O CONTROL BIT
      JSB $LIBX       * TURN ON INTERRUPTS AND MEMORY PROTECT
      DEF DT15B       * RETURN TO CALLING PROGRAM
*
STORE DEF DATA        * DEFINE DATA BUFFER ADDRESS
*
      END DT15B
```

```
ASMB,L
*
*
      NAM WAIT1
*
*****************************************************
*                                                   *
*     THIS SUBROUTINE PROVIDES A WAITING LOOP. THE  *
*     VARIABLE "DELAY" DETERMINES THE LENGTN OF     *
*     TIME WAITING.  IT IS A NUMBER FROM 0 TO       *
*     32767.                                        *
*                                                   *
*****************************************************
*
      ENT WAIT
      COM DELAY
*
WAIT  NOP
      LDY DELAY     * SET TIMER
      ISY           * INCREMENT TIMER ONCE BEFORE DECREMENTING
      NOP
*
LOOP  DSY           * DECREMENT COUNT. TIMER=0?
      JMP *-1       * NO. CONTINUE
      JMP WAIT,I    * YES. RETURN TO CALLING ROUTINE.
*
      END WAIT
```

```
ASMB,L
*
*
      NAM AD14
*
*****************************************************
*                                                   *
*     THIS SUBROUTINE IS VERY SIMILAR TO AD14B.     *
*     ITS PURPOSE IS TO PROVIDE AN APPROXIMATION    *
*     OF THE CONVERSION TIME PLUS THE TIME REQUIRED *
*     TO STORE DATA WHICH OCURRS IN AD14B.  AD14    *
*     GOES INTO AN INFINITE LOOP WHICH CAN BE TERM- *
*     INATED BY PRESSING THE "CLEAR DISPLAY"        *
*     BUTTON ON THE FRONT PANEL.  WHILE THIS ROU-   *
*     TINE IS RUNNING, THE "STATUS" SIGNAL ON THE   *
*     A/D CONVERTER CAN BE MONITERED WITH AN OSCIL- *
*     LOSCOPE.                                      *
*                                                   *
*     VARIABLE:                                     *
*       DELAY - ADDS DELAY TIME BETWEEN THE ACQUI-  *
*               SITION OF EACH TWO DATA POINTS      *
*                                                   *
*****************************************************
*
      EXT $LIBR,$LIBX,WAIT
      ENT DT14B
*
      COM DELAY
*
DT14B NOP
      JSB $LIBR       * TURN OFF INTERRUPTS AND MEMORY PROTECT
      NOP             *
      LDA 1           * SET THE SWITCH
      OTA 1           *   REGISTER TO 1
      LDA DELAY       *
      SSA             * SKIP IF DELAY>0 OR =0
      CMA             * COMPLEMENT IF DELAY<0
      STA DELAY       *
      LDX BUFL        * SET COUNTER
      LDB STORE       *
*
LOOP  STC 14B,C       * SEND ENCODE SIGNAL
      SFS 14B         * IS DEVICE BUSY?
      JMP *-1         * YES. TRY AGAIN.
      LIA 14B         * NO. READ DATA.
      STA SAVE        *
      STB STORE       *
      JSB WAIT        * DELAY PERIOD BEFORE GETTING NEW DATA
      LIB 1           * LOAD REG. B WITH SWITCH REG.
      SZB             * SKIP WHEN SWITCH REG. IS CLEAR
      JMP LOOP        * GET NEXT DATA
*
      CLC 14B,C       * CLEAR I/O CONTROL BIT
      JSB $LIBX       * TURN ON INTERRUPTS AND MEMORY PROTECT
      DEF DT14B       * RETURN TO CALLING PROGRAM
```

```
*
STORE EQU 0          * DUMMY VARIABLE
SAVE  EQU 0          * DUMMY VARIABLE
BUFL  EQU 0          * DUMMY VARIABLE
*
      END DT14B

ASMB,L
*
*
      NAM AD15
*
*****************************************************
*                                                   *
*     THIS SUBROUTINE IS VERY SIMILAR TO AD15B.     *
*     ITS PURPOSE IS TO PROVIDE AN APPROXIMATION    *
*     OF THE CONVERSION TIME PLUS THE TIME REQUIRED *
*     TO STORE DATA WHICH OCURRS IN AD15B.  AD15    *
*     GOES INTO AN INFINITE LOOP WHICH CAN BE TERM- *
*     INATED BY PRESSING THE "CLEAR DISPLAY"        *
*     BUTTON ON THE FRONT PANEL.  WHILE THIS ROU-   *
*     TINE IS RUNNING, THE "STATUS" SIGNAL OF THE   *
*     A/D CONVERTER CAN BE MONITERED WITH AN OSCIL- *
*     LOSCOPE.                                      *
*                                                   *
*     VARIABLE:                                     *
*       DELAY - ADDS DELAY TIME BETWEEN THE ACQUI-  *
*               SITION OF EACH TWO DATA POINTS      *
*                                                   *
*****************************************************
*
      EXT $LIBR,$LIBX,WAIT
      ENT DT15B
*
      COM DELAY
*
DT15B NOP
      JSB $LIBR      * TURN OFF INTERRUPTS AND MEMORY PROTECT
      NOP            *
      LDA 1          * SET THE SWITCH
      OTA 1          *   REGISTER TO 1
      LDA DELAY      *
      SSA            * SKIP IF DELAY>0 OR =0
      CMA            * COMPLEMENT IF DELAY<0
      STA DELAY      *
      LDX BUFL       * SET COUNTER
      LDB STORE      *
*
LOOP  STC 15B,C      * SEND ENCODE SIGNAL
      SFS 15B        * IS DEVICE BUSY?
      JMP *-1        * YES. TRY AGAIN.
      LIA 15B        * NO. READ DATA.
      STA SAVE       *
      STB STORE      *
      JSB WAIT       * DELAY PERIOD BEFORE GETTING NEW DATA
```

```
      LIB 1          * LOAD REG. B WITH SWITCH REG.
      SZB            * SKIP WHEN SWITCH REG. IS CLEAR
      JMP LOOP       * GET NEXT DATA
*
      CLC 15B,C      * CLEAR I/O CONTROL BIT
      JSB $LIBX      * TURN ON INTERRUPTS AND MEMORY PROTECT
      DEF DT15B      * RETURN TO CALLING PROGRAM
*
STORE EQU 0          * DUMMY VARIABLE
SAVE  EQU 0          * DUMMY VARIABLE
BUFL  EQU 0          * DUMMY VARIABLE
*
      END DT15B
```

```
FTN4,L
C
C
      PROGRAM ANDG2
C
C*****************************************************************
C*                                                               *
C*    THIS PROGRAM TESTS THE SUBROUTINE DMAIN.  THE              *
C*    ENTRY POINTS ARE CHK6B, DT14B, AND DT15B.                  *
C*                                                               *
C*****************************************************************
C
      INTEGER DATA,BUFL,DECIDE,FNSH6
      COMMON DATA(8192),BUFL,FNSH6
C
C---  INITIALIZE VARIABLES
C
      WRITE(1,200)
      READ(1,*) BUFL
      BUFL = -BUFL
      WRITE(1,300)
      READ(1,*) LUNUM
      WRITE(1,400)
      READ(1,*) DECIDE
C
C---  CHECK IF DCPC CHAN. 1 IS AVAILABLE
C
      FNSH6 = 0
100   CALL CHK6B
      IF(FNSH6.EQ.0) GO TO 100
C
C---  GET THE DATA
C
      IF(DECIDE.EQ.14) CALL DT14B
      IF(DECIDE.EQ.15) CALL DT15B
C
C---  WAIT UNTIL DATA ACQUISITION IS COMPLETE
C
      FNSH6 = 0
150   CALL CHK6B
      IF(FNSH6.EQ.0) GO TO 150
C
C---  OUTPUT DATA
C
      BUFL = -BUFL
      WRITE(LUNUM,500) (I,DATA(I),I=1,BUFL)
C
200   FORMAT(//,"HOW MANY DATA POINTS?  (8192 MAX) ")
C
300   FORMAT(//,"WHAT IS THE LU# OF THE OUTPUT DEVICE? ")
C
400   FORMAT(//,"WHICH I/O CHANNEL, 14 OR 15?")
C
500   FORMAT(22(/),128(4(2I7),/))
C
```

```
      STOP
      END
ASMB,L
*
*
      NAM DMAIN
*
**************************************************************
*                                                            *
*     THIS SUBROUTINE ACQUIRES DATA FROM AN A/D CONVERTER   *
*     USING DIRECT MEMORY ACCESS.  ONE OF TWO CHANNELS      *
*     CAN BE SELECTED: 14 OR 15.                             *
*                                                            *
**************************************************************
*
      EXT $LIBR,$LIBX
      ENT CHK6B,DT14B,DT15B
*
      COM DATA(8192),BUFL,FNSH6
*
CHK6B NOP               * ENTRY POINT
      JSB $LIBR         * TURN OFF INTERRUPTS AND MEMORY PROTECT
      NOP               *
      SFS 6B            * IS DCPC CHAN. 1 AVAILABLE?
      JMP RTN           * NO. RETURN.
      LDA =B1           * YES.
      STA FNSH6         *  SET FLAG WORD TO 1.
RTN   JSB $LIBX         * TURN ON INTERRUPTS AND MEMORY PROTECT
      DEF CHK6B         * RETURN TO CALLING PROGRAM
*
DT14B NOP               * ENTRY POINT
      JSB $LIBR         *
      NOP               *
      LDA BUFL          *
      STA CW3           * SAVE -BUFL
      LDA CW2           * SPECIFY THAT
      ADA =B100000      *  OPERATION IS INPUT
      STA CW2           *
*
      LDA CW114         *
      OTA 6B            * SEND CW1 TO DCPC CHAN. 1
      CLC 2B            * PREPARE M REG. TO RECEIVE CW2
      LDA CW2           *
      OTA 2B            * SEND CW2 TO DCPC CHAN. 1
      STC 2B            * PREPARE FOR CW3
      LDA CW3           *
      OTA 2B            * SEND CW3 TO DCPC CHAN. 1
      STC 6B,C          * ACTIVATE DCPC CHAN. 1
      STC 14B,C         * START A/D CONVERSION
      CLC 6B,C          * DISABLE DCPC CHAN. 1 INTERRUPT
      CLC 14B,C         * DISABLE I/O CHAN. 14 INTERRUPT
      JSB $LIBX         *
      DEF DT14B         * RETURN TO CALLING PROGRAM
*
```

```
DT15B NOP            * ENTRY POINT
      JSB $LIBR      *
      NOP            *
      LDA BUFL       *
      STA CW3        *
      LDA CW2        *
      ADA =B100000   *
      STA CW2        *
*
      LDA CW115      *
      OTA 6B         *
      CLC 2B         *
      LDA CW2        *
      OTA 2B         *
      STC 2B         *
      LDA CW3        *
      OTA 2B         *
      STC 6B,C       *
      STC 15B,C      *
      CLC 6B,C       *
      CLC 15B,C      * DISABLE I/O CHAN. 15 INTERRUPT
      JSB $LIBX      *
      DEF DT15B      *
*
CW114 OCT 120014     *
CW115 OCT 120015     *
CW2   DEF DATA       *
CW3   BSS 1          *
*
      END DT14B
```

```
ASMB,L
*
      NAM DVR50
*
************************************************************
*
*     THIS IS A DUMMY DRIVER.  ITS ONLY FUNCTION IS TO
*     JUMP TO A USER SUPPLIED SUBROUTINE WHICH PERFORMS
*     THE NECESSARY DRIVER FUNCTIONS.  THIS PROCEDURE ALLOWS
*     TROUBLESHOOTING OF THE DRIVER AND CHANGES TO BE MADE
*     WITHOUT REGENERATING THE OPERATING SYSTEM.  A LARGE
*     BUFFER AREA IS INCLUDED SO THAT THE CORRECTED VERSION
*     CAN EVENTUALLY BE COPIED INTO THE SYSTEM AREA.
*
************************************************************
*
*
      ENT DR50,I.50,C.50
*
*
I.50  NOP              * INITIATION SECTION *
      JSB DR50,I
      JMP I.50,I
*
C.50  NOP              * CONTINUATION COMPLETION SECTION *
      JSB DR50,I
      JMP C.50,I
*
DR50  DEF TMP50        * THE USER'S PROGRAM MUST CHANGE TMP50
*                      * STARTING ADDRESS OF HIS DRIVER SUBROU
*
TMP50 NOP
      JMP TMP50,I
      BSS 300B
      END

ASMB,L
*
      NAM DVR51
*
************************************************************
*
*     THIS IS A DUMMY DRIVER.  ITS ONLY FUNCTION IS TO
*     JUMP TO A USER SUPPLIED SUBROUTINE WHICH PERFORMS
*     THE NECESSARY DRIVER FUNCTIONS.  THIS PROCEDURE ALLOWS
*     TROUBLESHOOTING OF THE DRIVER AND CHANGES TO BE MADE
*     WITHOUT REGENERATING THE OPERATING SYSTEM.  A LARGE
*     BUFFER AREA IS INCLUDED SO THAT THE CORRECTED VERSION
*     CAN EVENTUALLY BE COPIED INTO THE SYSTEM AREA.
*
************************************************************
*
*
      ENT DR51,I.51,C.51
*
```

```
*
I.51   NOP              * INITIATION SECTION *
       JSB DR51,I
       JMP I.51,I
*
C.51   NOP              * CONTINUATION COMPLETION SECTION *
       JSB DR51,I
       JMP C.51,I
*
DR51   DEF TMP51        * THE USER'S PROGRAM MUST CHANGE TMP51
*                       * STARTING ADDRESS OF HIS DRIVER SUBROU
*
TMP51  NOP
       JMP TMP51,I
       BSS 300B
       END

ASMB,L
*
       NAM DVR52
*
**************************************************************
*
*      THIS IS A DUMMY DRIVER.  ITS ONLY FUNCTION IS TO
*      JUMP TO A USER SUPPLIED SUBROUTINE WHICH PERFORMS
*      THE NECESSARY DRIVER FUNCTIONS.  THIS PROCEDURE ALLOWS
*      TROUBLESHOOTING OF THE DRIVER AND CHANGES TO BE MADE
*      WITHOUT REGENERATING THE OPERATING SYSTEM.  A LARGE
*      BUFFER AREA IS INCLUDED SO THAT THE CORRECTED VERSION
*      CAN EVENTUALLY BE COPIED INTO THE SYSTEM AREA.
*
**************************************************************
*
*
       ENT DR52,I.52,C.52
*
*
I.52   NOP              * INITIATION SECTION *
       JSB DR52,I
       JMP I.52,I
*
C.52   NOP              * CONTINUATION COMPLETION SECTION *
       JSB DR52,I
       JMP C.52,I
*
DR52   DEF TMP52        * THE USER'S PROGRAM MUST CHANGE TMP52
*                       * STARTING ADDRESS OF HIS DRIVER SUBROU
*
TMP52  NOP
       JMP TMP52,I
       BSS 300B
       END
```

Appendix F: System User's Guide

This appendix contains supplementary information which should assist the user in getting acquainted with the DSP laboratory. It is not intended to be comprehensive because the user can find most information in appropriate manuals or other documentation.

1. The system master security code is AF.
2. Off-line loading procedures

   a. System Disk

   First, make sure the computer has halted, and the "DRIVE READY" light on the disk drive is on. Set the S Reg. to 111202 (the disk loader ROM occupies optional loader ROM socket #2; Ref 14:7). Press the keys PRESET, IBL, and RUN in sequence. This will load the operating system into memory. The system will come up with FMGR running.

   b. HP Cassettes

   Select the left or right cassette drive by pressing GOLD, one of the FROM: keys, and TRANSMIT in sequence. Then, follow the same procedure as described for the disk, but set the S Reg. to 041300 (optional loader ROM #1).

   c. Paper Tape Reader

   Follow the same procedure as described for the disk, but set the S Reg. to 001700.

3. Compile and run a FORTRAN IV program

   To compile a program, enter the following command:

```
*RU,FTN4,&fname,1,-
```

The asterisk is the RTE prompt. The source code file name is &fname. HP convention is to start all source code files with a "&" and all object files with a "%". For the above command, if no file exists, called %fname, then RTE will create the file, stop the compilation, and report a DISASTER 97. In this case, just type the same command in again, and it should run. If %fname existed before the command was entered, the new object code will overwrite it as the compilation proceeds.

To load and run the program, enter the following commands (the colon is the FMGR prompt):

```
*RU,FMGR
:LG,0
:MR,%fname
:RU,LOADR,99,1
```

If no errors were discovered by the loader, it will report that the program is ready. To run the program, type:

```
:RU,pname
```

where pname is the name of the program. Usually the FMGR prompt will return before all of the output has been sent to the designated device. To see the remaining output, strike the RETURN key. This will also terminate FMGR.

A program can also be run without FMGR so that no interruption in the output occurs. Just enter the EX command after running the loader. Then type:

```
*RU,pname
```

4. Assemble and run an assembly language program.

   Enter the following commands:

```
*RU,FMGR
:LS,0
:MS,&fname
:LG,0
:LG,3
:EX
*RU,ASMB,2,1,99
*RU,FMGR
:SA,LG,%fname
```

   The program can now be loaded and run in the same way as described for FORTRAN IV, except that the MR command is not required.

5. Report Files

   The Source code file name of any program listed in this report is just the program name preceeded by a "&". The corresponding object code files are preceeded by a "%". Any corresponding type 6 file has the same name as the program name.

6. TI Terminal

   The TI terminal is currently being operated at 110 baud. A 300 baud interface is available, but the terminal has trouble completing a carriage return fast enough. Another printing terminal has been successfully operated on the same I/O channel using the 300 baud interface.

Appendix G: Disk Storage Programs

The two programs implemented for storing data on the disk are listed in this Appendix. STDSK demonstrates temporary disk storage. FILES demonstrates permanent disk storage in a disk file. Both programs are self-documenting and require no further explanation.

```
FTN4,L
C
C
      PROGRAM STDSK
C
C
C*************************************************************
C
C     THIS PROGRAM USES A DISK TRACK ALLOCATION EXEC CALL TO
C     ALLOCATE TRACKS FOR DATA STORAGE.  THE SUBROUTINE BINRY1
C     STORES DATA ON THE DISK.
C
C     VARIABLES:
C       IBUFR - THIS IS THE DATA BUFFER TO BE READ FROM OR
C               WRITTEN TO
C       IBUFL - LENGTH OF THE BUFFER
C       IDISK - LOGICAL UNIT NUMBER OF THE DISK USED IN THE
C               DATA TRANSFER
C       ISTRK - DISK STARTING TRACK NUMBER
C       ISCTR - DISK STARTING SECTOR NUMBER
C       IOFST - DISK SECTOR OFFSET (I.E., THE NUMBER OF WORDS TO
C               TO BE SKIPPED IN THE SECTOR)
C       IRDWR - DETERMINES READ OR WRITE OPERATION
C               1 = READ    2 = WRITE
C       ICODE - SPECIFIES THE TYPE OF EXEC CALL
C
C*************************************************************
C
C
      DIMENSION IBUFR(100)
C
C
C---  ALLOCATE TRACKS
C
      WRITE(1,50)
50    FORMAT(//,"HOW MANY TRACKS?")
      READ(1,*) ITRAK
      ICODE = 4
      CALL EXEC(ICODE,ITRAK,ISTRK,IDISK,ISECT)
C
C---  INFORM USER OF DATA STORAGE AREA, OR THAT IT IS
C---  NOT AVAILABLE
C
      IF(ISTRK.EQ.-1) CALL EXIT
      WRITE(1,75) ISTRK,IDISK,ISECT
75    FORMAT(//,"THE STARTING TRACK NUMBER IS",I4,/,
     +"THE DISK LU NUMBER IS",I4,/,
     +"THE NUMBER OF 64 WORD SECTORS/TRACK IS",I4,//)
C
C---  SET VARIABLES FOR BINRY
C
      DO 100 I=1,100
100   IBUFR(I) = I
      IBUFL = 100
      ISCTR = 0
```

```
      IOFST = 0
      IRDWR = 2
C
C---  WRITE THE BUFFER VALUES TO THE DISK
C
      CALL BNRY1(IBUFR,IBUFL,IDISK,ISTRK,ISCTR,IOFST,IRDWR)
C
C---  SET THE BUFFER VALUES TO ZERO
C
      DO 200 I=1,100
200   IBUFR(I) = 0
C
C---  READ THE BUFFER VALUES BACK IN FROM THE DISK
C
      IRDWR = 1
      CALL BNRY1(IBUFR,IBUFL,IDISK,ISTRK,ISCTR,IOFST,IRDWR)
C
C---  WRITE OUT THE BUFFER VALUES TO VERIFY THE READ/WRITE
C---  OPERATIONS
C
      WRITE(1,300) (IBUFR(I),I=1,100)
300   FORMAT("BUFFER VALUES",//,10(10I5,/))
C
      STOP
      END
C
C
C
C
      SUBROUTINE BNRY1(IBUFR,IBUFL,IDISK,ISTRK,ISCTR,IOFST,IRDWR)
C
C
C
C*************************************************************
C
C     THIS ROUTINE USES THE SYSTEM UTILITY "BINRY" FOR READING
C     FROM AND WRITTING TO A DISK.  THE ENTRY POINT FOR READING
C     IS "BREAD".  THE ENTRY POINT FOR WRITTING IS "BWRIT".
C
C*************************************************************
C
C
C
      DIMENSION IBUFR(IBUFL)
C
C
C
      IF(IRDWR.EQ.1) GO TO 100
      IF(IRDWR.EQ.2) GO TO 200
      WRITE(1,50)
50    FORMAT("***** ERROR IN INPUT.  IRDWR DOES NOT EQUAL ",
     +        "1 OR 2 *****")
      RETURN
100   CALL BREAD(IBUFR,IBUFL,IDISK,ISTRK,ISCTR,IOFST)
      RETURN
200   CALL BWRIT(IBUFR,IBUFL,IDISK,ISTRK,ISCTR,IOFST)
      RETURN
      END
C
```

```
C
C
C
      SUBROUTINE EXIT
C
C
C
C*****************************************************************
C
C     THIS SUBROUTINE IS CALLED WHEN THE TRACKS REQUESTED ARE
C     UNAVAILABLE.  THE USER IS INFORMED OF THIS AND THE PROGRAM
C     IS STOPPED.
C
C*****************************************************************
C
C
C
      WRITE(1,1000)
1000  FORMAT(//,"TRACK(S) ARE NOT AVAILABLE.",
     +"PROGRAM TERMINATED.",//)
      STOP
      END
```

```
FTN4,L
C
C
      PROGRAM FILES
C
C*************************************************************
C*                                                          *
C*    THIS PROGRAM DEMONSTRATES THE USE OF FMP CALLS FOR    *
C*    DATA STORAGE ON A DISK. A FILE IS CREATED, AND DATA   *
C*    IS WRITTEN INTO AND READ FROM THE FILE.               *
C*                                                          *
C*************************************************************
C
C
      INTEGER DCB1(144),NAME1(3),SIZE1(2),CR,TYPE1,SECUR,
     +        ERROR,OPTN,ARAY1(10),ARAY2(10),ARAY3(10)
C
      NAME1(1) = 0
      NAME1(2) = 0
      NAME1(3) = 0
      WRITE(1,25)
25    FORMAT(/"ENTER FILE NAME")
      READ(1,50) (NAME1(I),I=1,3)
50    FORMAT(3A2)
      SIZE1(1) = 3
      TYPE1 = 3
      SECUR = 0
      CR = 20
C
C---  CREATE A TYPE 3 FILE ON CARTRIDGE 20
C
      CALL CREAT(DCB1,ERROR,NAME1,SIZE1,TYPE1,SECUR,CR)
      IF(ERROR.LT.0) GO TO 900
C
      WRITE(1,100) (NAME1(I),I=1,3),CR,ERROR
100   FORMAT(/"FILE <",3A2,"> HAS BEEN CREATED ON CARTRIDGE ",
     +       I2," WITH ",I3," 64-WORD SECTORS"/)
C
C---  OPEN THE FILE FOR READ/WRITE
C
      OPTN = 0
      CALL OPEN(DCB1,ERROR,NAME1,OPTN,SECUR,CR)
      IF(ERROR.LT.0) GO TO 900
C
C---  GENERATE DATA INTO THREE ARRAYS
C
      DO 200 I=1,10
        ARAY1(I) = I
        ARAY2(I) = 2*I
        ARAY3(I) = I*I
200   CONTINUE
C
C---  WRITE EACH DATA ARRAY AS ONE RECORD IN THE FILE
C
      LENGH = 10
```

```
      CALL WRITF(DCB1,ERROR,ARRAY1,LENGH)
      IF(ERROR.LT.0) GO TO 900
      CALL WRITF(DCB1,ERROR,ARRAY2,LENGH)
      IF(ERROR.LT.0) GO TO 900
      CALL WRITF(DCB1,ERROR,ARRAY3,LENGH)
      IF(ERROR.LT.0) GO TO 900
C
C---  SET DATA ARRAYS TO ZERO
C
      DO 300 I=1,10
        ARRAY1(I) = 0
        ARRAY2(I) = 0
        ARRAY3(I) = 0
300   CONTINUE
C
C---  READ EACH RECORD FROM FILE INTO AN ARRAY
C
      CALL RWNDF(DCB1,ERROR)
      IF(ERROR.LT.0) GO TO 900
C
      CALL READF(DCB1,ERROR,ARRAY1,LENGH)
      IF(ERROR.LT.0) GO TO 900
      CALL READF(DCB1,ERROR,ARRAY2,LENGH)
      IF(ERROR.LT.0) GO TO 900
      CALL READF(DCB1,ERROR,ARRAY3,LENGH)
      IF(ERROR.LT.0) GO TO 900
C
C---  WRITE THE DATA ARRAYS TO VERIFY READ/WRITE OPERATIONS
C
      WRITE(1,400)  (ARRAY1(I),I=1,10),(ARRAY2(I),I=1,10),
     +               (ARRAY3(I),I=1,10)
400   FORMAT(/3(10I7/)/)
C
C---  CLOSE DATA FILE
C
      CALL CLOSE(DCB1,ERROR)
      IF(ERROR.LT.0) GO TO 900
C
      STOP
C
900   WRITE(1,1000) ERROR
1000  FORMAT(/"FMP ERROR ",I4/)
C
      STOP
      END
```

Appendix H: Auto Plot Program

This Appendix contains a listing of the program APLOT. It is self-documenting.

```
FTN4,L
C
C
      PROGRAM APLOT
C
C*************************************************************
C*                                                          *
C*    THIS PROGRAM DEMONSTRATES COMPUTER CONTROL OF THE     *
C*    GRAPHICS TERMINAL.  THE EXECUTIVE CALL IS USED TO     *
C*    SEND THE DECIMAL VALUE OF THE ASCII CHARACTERS.       *
C*    THE SEQUENCE OF CHARACTERS LISTED IN BUFR WILL        *
C*    FILL THE AUTO PLOT MENU, ACCEPT DATA FROM THE COM-    *
C*    PUTER AND PLOT IT, AND TURN OFF AUTO PLOT.            *
C*                                                          *
C*    VARIABLES:                                            *
C*      CODE - DESIGNATES A WRITE EXEC CALL                 *
C*      CTLWD - SPECIFIES THE SYSTEM CONSULE LU#            *
C*      BUFR - CHARACTER BUFFER                             *
C*      BUFL - BUFFER LENGTH; = -2*(# OF WORDS IN BUFR)     *
C*                                                          *
C*************************************************************
C
      INTEGER CODE,CTLWD,BUFR(43),BUFL
C
C---  INITIALIZE VARIABLES
C
      CODE = 2
      CTLWD = 1
      BUFL = -86
      BUFR(1) = 27
      BUFR(2) = 42
      BUFR(3) = 97
      BUFR(4) = 100
      BUFR(5) = 50
      BUFR(6) = 104
      BUFR(7) = 49
      BUFR(8) = 105
      BUFR(9) = 50
      BUFR(10) = 106
      BUFR(11) = 49
      BUFR(12) = 107
      BUFR(13) = 48
      BUFR(14) = 108
      BUFR(15) = 49
      BUFR(16) = 48
      BUFR(17) = 48
      BUFR(18) = 109
      BUFR(19) = 48
      BUFR(20) = 110
      BUFR(21) = 49
      BUFR(22) = 48
```

```
      BUFR(23) = 48
      BUFR(24) = 111
      BUFR(25) = 50
      BUFR(26) = 48
      BUFR(27) = 112
      BUFR(28) = 49
      BUFR(29) = 48
      BUFR(30) = 113
      BUFR(31) = 50
      BUFR(32) = 48
      BUFR(33) = 114
      BUFR(34) = 49
      BUFR(35) = 48
      BUFR(36) = 115
      BUFR(37) = 49
      BUFR(38) = 52
      BUFR(39) = 117
      BUFR(40) = 49
      BUFR(41) = 118
      BUFR(42) = 79
      BUFR(43) = 65
C
C---  SET UP AUTO PLOT MENU AND DRAW GRID
C
      CALL EXEC(CODE,CTLWD,BUFR,BUFL)
C
C---  SEND DATA TO BE PLOTTED
C
      J = 1
      K = 0
      DO 200 I=1,92,7
        WRITE(1,100) I,J
100     FORMAT(2I7)
        K = K+1
        J = J+K
200   CONTINUE
C
C---  TURN OFF AUTO PLOT
C
      BUFL = -8
      BUFR(4) = 66
      CALL EXEC(CODE,CTLWD,BUFR,BUFL)
C
      STOP
      END
```

Appendix I: D/A Converter Software Listings

The program TSTDA generates data for subroutine DA16B and calls it. DA16B provides software support for the D/A converter which has an X-Y plotter and a conventional oscilloscope connected in parallel to its outputs. Subroutines MINXY, MAXXY, SCALE, CBINE, PLOTR, and SCOPE are called by DA16B. Subroutine OSCPE (entry point SCPE) is called by SCOPE. Subroutine PLT (entry point PLOT) and ONE (entry point FIRST) are called by PLOTR. The purpose of each subroutine is explained in the program listing.

```
FTN4,L
C
C
      PROGRAM TSTDA
C
C*************************************************************
C*                                                          *
C*    THIS PROGRAM GENERATES DATA IN SAMPLE POINT FORM      *
C*    FOR A CONTINUOUS FUNCTION.  THE DATA IS USED TO       *
C*    TO DEMONSTRATE THE SUBROUTINE 'DA16B'.                *
C*                                                          *
C*************************************************************
C
      COMMON NPTS,IXAXS(256),JYAXS(256),XAXIS(256),YAXIS(256)
C
C---  INPUT THE NUMBER OF DATA POINTS
C
      WRITE(1,50)
50    FORMAT(22(/),"HOW MANY DATA POINTS?   (256 MAX)")
      READ(1,*) NPTS
C
C---  GENERATE THE DATA
C
      PI = 4.0*ATAN(1.0)
      STEP = 3.0*PI/FLOAT(NPTS)
C
      DO 100 I=1,NPTS
        SAVE = FLOAT(I)*STEP
        XAXIS(I) = SAVE
        YAXIS(I) = SIN(SAVE)+(1.0/SAVE)*COS(SAVE)
100   CONTINUE
C
C---  SEND DATA TO D/A AND OUTPUT DEVICE
C
      CALL DA16B
C
      STOP
      END
FTN4,L
C
C
      SUBROUTINE DA16B
C
C*************************************************************
C*                                                          *
C*    THIS ROUTINE FORMATS DATA FOR OUTPUT TO EITHER AN     *
C*    X-Y PLOTTER OR A CONVENTIONAL OSCILLOSCOPE.  THE      *
C*    INPUTS ARE TWO REAL ARRAYS CORRESPONDING TO THE X     *
C*    AND Y AXES, AND THE NUMBER OF DATA POINTS.            *
C*                                                          *
C*************************************************************
C
      COMMON NPTS,IXAXS(256),JYAXS(256),XAXIS(256),YAXIS(256)
```

```
C
C---  FIND THE MINIMUM VALUES IN XAXIS AND YAXIS
C
      CALL MINXY(XMIN,YMIN)
C
C---  SET XMIN AND YMIN TO MAKE ALL ARRAY
C---  VALUES POSITIVE
C
      IF(XMIN.GT.0.0) XMIN = 0.0
      IF(YMIN.GT.0.0) YMIN = 0.0
      XMIN = ABS(XMIN)
      YMIN = ABS(YMIN)
C
C---  FIND THE MAXIMUM VALUES IN XAXIS AND YAXIS
C
      CALL MAXXY(XMAX,YMAX)
C
C---  SCALE ALL ARRAY VALUES
C
      CALL SCALE(XMIN,YMIN,XMAX,YMAX)
C
C---  COMBINE INTEGER ARRAYS INTO ONE ARRAY
C
      WRITE(1,100)  (IXAXS(I),JYAXS(I),I=1,NPTS)
100   FORMAT(//,37(14I5,/),//)
      CALL CBINE
C
C---  SELECT OUTPUT DEVICE:  PLOTTER OR OSCILLOSCOPE
C
      WRITE(1,150)
150   FORMAT(//,"WHAT TYPE OF OUTPUT DEVICE?",/,
     +          "1=PLOTTER   2=OSCILLOSCOPE")
      READ(1,*) IPICK
C
C---  OUTPUT DATA
C
      IF(IPICK.EQ.1) CALL PLOTR
      IF(IPICK.EQ.2) CALL SCOPE
C
      RETURN
      END
C
C
C
C
      SUBROUTINE MINXY(XMIN,YMIN)
C
C---  THIS ROUTINE DETERMINES THE MINIMUM VALUE FOR EACH AXIS
C
      COMMON NPTS,IXAXS(256),JYAXS(256),XAXIS(256),YAXIS(256)
C
      XMIN = XAXIS(1)
      YMIN = YAXIS(1)
C
      DO 200 I=2,NPTS
        IF(XAXIS(I).LT.XMIN) XMIN = XAXIS(I)
```

```
        IF(YAXIS(I).LT.YMIN) YMIN = YAXIS(I)
300   CONTINUE
C
      RETURN
      END
C
C
C
C
      SUBROUTINE MAXXY(XMAX,YMAX)
C
C---  THIS ROUTINE DETERMINES THE MAXIMUM VALUE FOR EACH AXIS
C
      COMMON NPTS,IXAXS(256),JYAXS(256),XAXIS(256),YAXIS(256)
C
      XMAX = XAXIS(1)
      YMAX = YAXIS(1)
C
      DO 300 I=2,NPTS
        IF(XAXIS(I).GT.XMAX) XMAX = XAXIS(I)
        IF(YAXIS(I).GT.YMAX) YMAX = YAXIS(I)
300   CONTINUE
C
      RETURN
      END
C
C
C
C
      SUBROUTINE SCALE (XMIN,YMIN,XMAX,YMAX)
C
      COMMON NPTS,IXAXS(256),JYAXS(256),XAXIS(256),YAXIS(256)
C
C---  MAKE ALL ARRAY VALUES RANGE FROM 0 TO 255:  ONE BYTE
C---  INITIALIZE THE INTEGER ARRAYS
C
      DO 400 I=1,NPTS
        XAXIS(I) = (XMIN+XAXIS(I))/(XMIN+XMAX)*255
        IXAXS(I) = IFIX(XAXIS(I))
        YAXIS(I) = (YMIN+YAXIS(I))/(YMIN+YMAX)*255
        JYAXS(I) = IFIX(YAXIS(I))
400   CONTINUE
C
      RETURN
      END
C
C
C
C
      SUBROUTINE SCOPE
C
      INTEGER READY
C
      WRITE(1,500)
500   FORMAT(//,55(1H*),/,"HIT A CARRIAGE RETURN TO START THE ",
     + "SCOPE DISPLAY. ",/,"PRESS 'CLEAR DISPLAY' TO TERMINATE.",
     + /,55(1H*),//)
      READ(1,*) READY
```

```
      CALL SCPE
C
      RETURN
      END
C
C
C
      SUBROUTINE PLOTR
C
      INTEGER READY
C
C---  POSITION THE PLOTTER PEN BY SENDING THE FIRST POINT
C
      CALL FIRST
C
      WRITE(1,600)
600   FORMAT(//,55(1H*),/,"SET THE PLOTTER PEN DOWN.",/,
     +        "HIT A CARRIAGE RETURN WHEN READY TO PLOT.",
     +        /,55(1H*),//)
      READ(1,*) READY
C
      CALL PLOT
C
      WRITE(1,610)
610   FORMAT(//,25(1H*),/,"PLOT COMPLETE. LIFT PEN.",/,
     +           25(1H*),//)
C
      RETURN
      END

ASMB,L
*
*
      NAM CMBNE
*
****************************************************************
*                                                              *
*     THIS ROUTINE COMBINES TWO INTEGER ARRAYS INTO ONE.       *
*     JYAXS IS THE MOST SIGNIFICANT BYTE, AND IXAXS IS         *
*     THE LEAST SIGNIFICANT BYTE.                              *
*                                                              *
****************************************************************
*
      ENT CBINE
*
      COM NPTS,IXAXS(256),JYAXS(256)
*
CBINE NOP             * ENTRY POINT
      CLA             *
      CAX             * CLEAR THE BUFFER PTR.
*
LOOP  LAX JYAXS       * MOST SIGNIFICANT BYTE
      ALF,ALF         * ROTATE REG. A LEFT ONE BYTE
      AND =B177400    * MAKE SURE RIGHT BYTE IS CLEAR
      LBX IXAXS       * LEAST SIGNIFICANT BYTE
```

```
      STB TEMP       *
      ADA TEMP       * COMBINE THE TWO BYTES
      SAX IXAXS      *   INTO ONE WORD
      ISX            * SET PTR. TO NEXT BUFFER LOCATION
      CXA            *
      CPA NPTS       * COMPARE BUFFER PTR. TO NPTS
      JMP CBINE,I    * RETURN TO CALLING PROGRAM
      JMP LOOP       * CONTINUE
*
TEMP  BSS 1
*
      END CBINE

ASMB,L
*
*
      NAM OSCPE
*
*****************************************************
*                                                   *
*     THIS ROUTINE SENDS DATA FROM A BUFFER TO AN OSCIL-   *
*     LOSCOPE THROUGH A D/A CONVERTER.              *
*                                                   *
*****************************************************
*
      EXT $LIBR,$LIBX
      ENT SCPE
*
      COM NPTS,IXAXS(256)
*
SCPE  NOP            * ENTRY POINT
      JSB $LIBR      * TURN OFF INTERRUPTS AND MEMORY PROTECT
      NOP            *
      LDA =B1        *
      STA 1B         * SET THE SWITCH REGISTER TO 1
*
TRACE CLA            * SET
      CAX            *   THE
      LDY NPTS       *   COUNTERS
      CLF 16B        * START THE REFRESH TIMER
*
NEXT  LAX IXAXS      *
      OTA 16B        * SEND DATA POINT TO BE PLOTTED
      ISX            * INCREMENT BUFFER PTR.
      DSY            * DECREMENT BUFFER COUNTER
      JMP NEXT       *
      SFS 16B        *
      JMP *-1        *
      LIA 1B         * CHECK THE SWITCH REGISTER
      SZA            * IS IT CLEAR?
      JMP TRACE      * NO. OUTPUT BUFFER AGAIN.
      CLC 16B        * YES. CLEAR I/O CONTROL BIT
      JSB $LIBX      * TURN ON INTERRUPTS AND MEMORY PROTECT
      DEF SCPE       * RETURN TO CALLING PROGRAM
*
```

```
      END SOPE

ASMB,L
*
*
      NAM PLT
*
*****************************************************************
*                                                               *
*     THIS ROUTINE SEND DATA FROM A BUFFER TO AN X-Y            *
*     PLOTTER THROUGH A D/A CONVERTER.                          *
*                                                               *
*****************************************************************
      EXT $LIBR,$LIBX
      ENT PLOT
*
      COM NPTS,IXAXS(256)
*
PLOT  NOP             * ENTRY POINT
      JSB $LIBR       * TURN OFF INTERRUPTS AND MEMORY PROTECT
      NOP             *
*
TRACE CLA             * SET
      CAX             *   THE
      LDY NPTS        *   COUNTERS
*
NEXT  LAX IXAXS       * LOAD BUFFER VALUE
*
      LDB =B-32767    * SET WAIT TIMER
WAIT  OTA 16B         *
      OTA 16B         *
      INB,SZB         * INCREMENT COUNT. FINISHED WAITING?
      JMP WAIT        * NO. CONTINUE.
*
      ISX             * YES. UPDATE BUFFER POINTER
      DSY             * DECREMENT COUNT. LAST DATA?
      JMP NEXT        * NO. GET NEXT DATA POINT
*
      CLC 16B         * YES. CLEAR I/O CONTROL BIT
      JSB $LIBX       * TURN ON INTERRUPTS AND MEMORY PROTECT
      DEF PLOT        * RETURN TO CALLING PROGRAM
*
      END PLOT

ASMB,L
*
*
      NAM ONE
*
*****************************************************************
*                                                               *
*     THIS ROUTINE SENDS THE FIRST BUFFER VALUE TO AN           *
*     X-Y PLOTTER THROUGH A D/A CONVERTER.  THIS WILL           *
*     POSITION THE PEN SO THE USER CAN SET IT DOWN.             *
*                                                               *
```

```
************************************************************
*
      EXT $LIBR,$LIBX
      ENT FIRST
*
      COM NPTS,IXAXS(1)
*
FIRST NOP             * ENTRY POINT
      JSB $LIBR       * TURN OFF INTERRUPTS AND MEMORY PROTECT
      NOP             *
      LDA IXAXS       * LOAD FIRST BUFFER VALUE
      OTA 16B         * OUTPUT BUFFER VALUE
      CLC 16B         * CLEAR I/O CONTROL BIT
      JSB $LIBX       * TURN ON INTERRUPTS AND MEMORY PROTECT
      DEF FIRST       * RETURN TO CALLING PROGRAM
*
      END FIRST
```

Appendix J: Segmented Program

Program MAIN1 will load and execute one of three disk-resident segments. The user is requested to enter 1, 2, or 3 corresponding to SEGM1, SEGM2, or SEGM3, respectively.

```
FTN4,L
C
C
      PROGRAM MAIN1
C
C*************************************************************
C*                                                           *
C*    THIS PROGRAM DEMONSTRATES A METHOD FOR WRITING SEG-    *
C*    MENTED PROGRAMS.  IT USES THE PROGRAM SEGMENT LOAD     *
C*    EXEC CALL.  ONE OF THREE SEGMENTS CAN BE CALLED.       *
C*                                                           *
C*************************************************************
C
      INTEGER SEGM1(3),SEGM2(3),SEGM3(3)
C
      DATA SEGM1/2HSE,2HGM,1H1/
      DATA SEGM2/2HSE,2HGM,1H2/
      DATA SEGM3/2HSE,2HGM,1H3/
      DATA ICODE/8/
C
C---  ASK USER WHICH SEGMENT
C
50    WRITE(1,100)
100   FORMAT(/,"WHICH SEGMENT  1, 2, OR 3?")
      READ(1,*) ICHOSE
C
C---  LOAD AND EXECUTE SEGMENT
C
      IF(ICHOSE.EQ.1) CALL EXEC(ICODE,SEGM1)
      IF(ICHOSE.EQ.2) CALL EXEC(ICODE,SEGM2)
      IF(ICHOSE.EQ.3) CALL EXEC(ICODE,SEGM3)
C
C---  TERMINATE PROGRAM IF ENTRY IS INVALID
C
      WRITE(1,200)
200   FORMAT(///,"ONLY THREE SEGMENTS AVAILABLE.  ",
     +            "EXECUTIVE PROGRAM TERMINATED.",///)
C
      STOP
      END
FTN4,L
C
C
      PROGRAM SEGM1(5)
C
C*************************************************************
C*                                                           *
C*    THIS IS SEGMENT 1 OF PROGRAM MAIN1.                    *
C*                                                           *
C*************************************************************
C
      WRITE(1,100)
100   FORMAT(//,"SEGMENT 1 HAS BEEN LOADED FROM THE ",
     +"DISK AND EXECUTED",/)
```

```
C
C---  RETURN TO MAIN1
C
      CALL MAIN1
      END
FTN4,L
C
C
      PROGRAM SEGM2(5)
C
C****************************************************************
C*                                                              *
C*    THIS IS SEGMENT 2 OF PROGRAM MAIN1.                       *
C*                                                              *
C****************************************************************
C
      WRITE(1,100)
100   FORMAT(//,"SEGMENT 2 HAS BEEN LOADED FROM THE ",
     +"DISK AND EXECUTED",/)
C
C---  RETURN TO MAIN1
C
      CALL MAIN1
      END
FTN4,L
C
C
      PROGRAM SEGM3(5)
C
C****************************************************************
C*                                                              *
C*    THIS IS SEGMENT 3 OF PROGRAM MAIN1.                       *
C*                                                              *
C****************************************************************
C
      WRITE(1,100)
100   FORMAT(//,"SEGMENT 3 HAS BEEN LOADED FROM THE ",
     +"DISK AND EXECUTED",/)
C
C---  RETURN TO MAIN1
C
      CALL MAIN1
      END
```

Appendix K: Executive Programs

Two programs were implemented to demonstrate the system's capability for executing a large number of DSP applications programs by means of an executive program. SCHED demonstrates the use of the Program Schedule EXEC Call. The user has the option of specifying one of four types of scheduling. MESG1 demonstrates the use of the System Message utility, MESSS. MESSS can be used to execute any of the system commands. However, only the RU command is included in MESG1.

```
FTN4,L
C
C
      PROGRAM SCHED
C
C*************************************************************
C*                                                          *
C*    THIS PROGRAM DEMONSTRATES THE USE OF THE PROGRAM      *
C*    SCHEDULE EXEC CALL.  THE USER SPECIFIES THE TYPE      *
C*    OF SCHEDULING AND ONE OF THREE PROGRAMS TO BE         *
C*    SCHEDULED.                                            *
C*                                                          *
C*************************************************************
C
      DIMENSION NAME1(3),NAME2(3),NAME3(3)
C
      DATA NAME1/2HPR,2HOG,1H1/
      DATA NAME2/2HPR,2HOG,1H2/
      DATA NAME3/2HPR,2HOG,1H3/
C
C---  SELECT TYPE OF SCHEDULE
C
50    WRITE(1,100)
100   FORMAT(//,"SELECT TYPE OF SCHEDULE.",//,
     +          " 9=IMMEDIATE SCHEDULE, WITH WAIT",/,
     +          "10=IMMEDIATE SCHEDULE, NO WAIT",/,
     +          "23=QUEUE SCHEDULE, WITH WAIT",/,
     +          "24=QUEUE SCHEDULE, NO WAIT",/)
      READ(1,*) ICODE
      IF(ICODE.EQ.9)  GO TO 300
      IF(ICODE.EQ.10) GO TO 300
      IF(ICODE.EQ.23) GO TO 300
      IF(ICODE.EQ.24) GO TO 300
C
C---  INVALID INPUT
C
      WRITE(1,1000)
      GO TO 50
C
C---  SELECT PROGRAM
C
300   WRITE(1,400)
400   FORMAT(//,"SELECT PROGRAM TO BE RUN.",//,
     +          "1=PROG1",/,"2=PROG2",/,"3=PROG3",/)
      READ(1,*) IPROG
      GO TO (500,600,700) IPROG
C
C---  INVALID INPUT
C
450   WRITE(1,1000)
      GO TO 300
C
C---  SCHEDULE PROGRAM
C
500   CALL EXEC(ICODE,NAME1)
```

```
      GO TO 750
600   CALL EXEC(ICODE,NAME2)
      GO TO 750
700   CALL EXEC(ICODE,NAME3)
750   WRITE(1,800)
800   FORMAT(//,"DO YOU WANT TO SCHEDULE ANOTHER PROGRAM?",/,
     +               "1=YES        2=NO",/)
      READ(1,*) IPICK
      IF(IPICK.EQ.1) GO TO 50
C
1000  FORMAT(//,"***** INPUT ERROR *****",//)
C
      STOP
      END

FTN4,L
C
C
      PROGRAM PROG1
C
C****************************************************************
C*                                                              *
C*    THIS PROGRAM IS SCHEDULED BY THE PROGRAM "SCHED".         *
C*                                                              *
C****************************************************************
C
      WRITE(1,100)
100   FORMAT(//,"PROGRAM 1 HAS BEEN SCHEDULED AND RUN",//)
C
      STOP
      END

FTN4,L
C
C
      PROGRAM PROG2
C
C****************************************************************
C*                                                              *
C*    THIS PROGRAM IS SCHEDULED BY THE PROGRAM "SCHED".         *
C*                                                              *
C****************************************************************
C
      WRITE(1,100)
100   FORMAT(//,"PROGRAM 2 HAS BEEN SCHEDULED AND RUN",//)
C
      STOP
      END

FTN4,L
C
C
      PROGRAM PROG3
C
```

```
C****************************************************************
C*                                                              *
C*     THIS PROGRAM IS SCHEDULED BY THE PROGRAM "SCHED".        *
C*                                                              *
C****************************************************************
C
      WRITE(1,100)
100   FORMAT(//,"PROGRAM 3 HAS BEEN SCHEDULED AND RUN",//)
C
      STOP
      END
```

```
FTN4,L
C
C
      PROGRAM MESG1
C
C*********************************************************************
C*                                                                  *
C*    THIS PROGRAM DEMONSTRATES THE USE OF THE SYSTEM               *
C*    MESSAGE UTILITY (MESSS).  THE UTILITY ALLOWS                  *
C*    SYSTEM COMMANDS TO BE EXECUTED FROM A USER PROGRAM.           *
C*    IN THIS CASE, THE USER HAS A CHOICE OF RUNNING ONE            *
C*    OF THREE PROGRAM.                                             *
C*                                                                  *
C*********************************************************************
C
      INTEGER CMND1(8),CMND2(8),CMND3(8)
C
C---  CMND1 = "RU,PROG1"
C---  CMND2 = "RU,PROG2"
C---  CMND3 = "RU,PROG3"
C
C---  SELECT PROGRAM TO BE RUN
C
50    WRITE(1,100)
100   FORMAT(//,"SELECT A COMMAND",/,"1=RU,PROG1",/,
     +           "2=RU,PROG2",/,"3=RU,PROG3",/)
      READ(1,*) IPICK
      GO TO (200,300,400) IPICK
C
C---  RUN PROGRAM 'PROG1'
C
200   CMND1(1) = 2HRU
      CMND1(2) = 2H,P
      CMND1(3) = 2HRO
      CMND1(4) = 2HG1
      ICNT1 = 8
      I = MESSS(CMND1,ICNT1)
      IF(I.LT.0) WRITE(1,2000) (CMND1(J),J=1,8)
      IF(I.GE.0) WRITE(1,3000) (CMND1(J),J=1,4)
      GO TO 900
C
C---  RUN PROGRAM 'PROG2'
C
300   CMND2(1) = 2HRU
      CMND2(2) = 2H,P
      CMND2(3) = 2HRO
      CMND2(4) = 2HG2
      ICNT2 = 8
      I = MESSS(CMND2,ICNT2)
      IF(I.LT.0) WRITE(1,2000) (CMND2(J),J=1,8)
      IF(I.GE.0) WRITE(1,3000) (CMND2(J),J=1,4)
      GO TO 900
C
C---  RUN PROGRAM 'PROG3'
C
```

```
400   CMND3(1) = 2HRU
      CMND3(2) = 2H,P
      CMND3(3) = 2HRO
      CMND3(4) = 2HG3
      ICNT3 = 8
      I = MESSS(CMND3,ICNT3)
      IF(I.LT.0) WRITE(1,2000) (CMND3(J),J=1,8)
      IF(I.GE.0) WRITE(1,3000) (CMND3(J),J=1,4)
C
C---  ASK IF ANOTHER PROGRAM IS TO BE RUN
C
900   WRITE(1,4000)
      READ(1,*) IPICK
      IF(IPICK.EQ.1) GO TO 50
C
2000  FORMAT(//,8A2,//)
3000  FORMAT(//,"THE COMMAND '",4A2,"' HAS BEEN EXECUTED",//)
4000  FORMAT(//,"DO YOU WANT TO RUN ANOTHER PROGRAM?",/,
     +           "1=YES    2=NO",/)
C
      STOP
      END
```

Appendix L: Recommended Software Structure

The software structure recommended for the DSP laboratory includes the Program Schedule EXEC Call and segmented programs. Program EXECT (the executive) has five valid commands: TERM (terminate the program), PRGM1, PRGM2, PRGM3, and LIST (lists the valid commands). The programs PRGM1, PRGM2, and PRGM3 are identical except for the names of the segments they can call. Therefore, only PRGM1 and its segments - SUBR1, SUBR2, and SUBR3 - are listed.

```
FTN4,L
C
C
      PROGRAM EXECT
C
C************************************************************
C*                                                          *
C*    THIS IS AN EXECUTIVE PROGRAM WHICH WILL SCHEDULE      *
C*    DISK RESIDENT PROGRAMS FOR IMMEDIATE EXECUTION.       *
C*    THERE ARE FIVE VALID COMMANDS:  PRGM1, PRGM2, PRGM3,  *
C*    LIST, AND TERM.  THE FIRST FOUR WILL SCHEDULE         *
C*    PROGRAMS.  'TERM' CAUSES EXECT TO TERMINATE.          *
C*                                                          *
C************************************************************
C
      COMMON ICMND(3),ICODE
C
      ICODE = 9
      WRITE(1,100)
100   FORMAT(10(/),10X,"***** PROGRAM FOR DIGITAL SIGNAL ",
     +  "PROCESSING *****",11(/))
150   WRITE(1,200)
200   FORMAT(/"ENTER COMMAND")
      ICMND(1) = 0
      ICMND(2) = 0
      ICMND(3) = 0
      READ(1,400)  (ICMND(I),I=1,3)
400   FORMAT(3A2)
      IF((ICMND(1).EQ.2HTE).AND.(ICMND(2).EQ.2HRM))  STOP
      IF((ICMND(1).EQ.2HPR).AND.(ICMND(2).EQ.2HGM)
     +   .AND.(ICMND(3).EQ.1H1))  GO TO 500
      IF((ICMND(1).EQ.2HPR).AND.(ICMND(2).EQ.2HGM)
     +   .AND.(ICMND(3).EQ.1H2))  GO TO 500
      IF((ICMND(1).EQ.2HPR).AND.(ICMND(2).EQ.2HGM)
     +   .AND.(ICMND(3).EQ.1H3))  GO TO 500
      IF((ICMND(1).EQ.2HLI).AND.(ICMND(2).EQ.2HST))  GO TO 500
C
C---  INTERCEPT INVALID COMMANDS
C
      CALL ERROR
      GO TO 150
C
C---  SCHEDULE PROGRAM
C
500   CALL EXEC(ICODE,ICMND)
      GO TO 150
      END
C
C
      SUBROUTINE ERROR
C
      COMMON ICMND(3),ICODE
C
      WRITE(1,100)
100   FORMAT(/"***** INPUT ERROR *****")
```

```
      WRITE(1,200)
200   FORMAT("SELECT OPTION:"/"1 = STOP"/"2 = CONTINUE"/
     +        "3 = DISPLAY LIST OF COMMANDS")
      READ(1,*) IPICK
      IF(IPICK.EQ.2) RETURN
      IF(IPICK.EQ.3) GO TO 300
      STOP
300   WRITE(1,400)
400   FORMAT(//15X,"***** LIST OF COMMANDS *****"//,
     + "PRGM1",5X,"PRGM2",5X,"PRGM3",5X,"LIST",6X,"TERM"//)
      RETURN
      END

FTN4,L
C
C
      PROGRAM PRGM1
C
      COMMON/IFLAG/IFLAG1
      INTEGER  SUBR1(3),SUBR2(3),SUBR3(3)
C
      DATA SUBR1/2HSU,2HBR,1H1/
      DATA SUBR2/2HSU,2HBR,1H2/
      DATA SUBR3/2HSU,2HBR,1H3/
      DATA ICODE/8/,IBGIN/0/
C
      IF(IBGIN.NE.100) GO TO 100
      GO TO (50,51,52) IFLAG1
100   IBGIN = 100
      WRITE(1,200)
200   FORMAT(/"PRGM1 HAS BEEN SCHEDULED BY EXECT")
C
C---  LOAD AND EXECUTE FIRST SEGMENT
C
      CALL EXEC(ICODE,SUBR1)
C
C---  LOAD AND EXECUTE SECOND SEGMENT
C
50    CALL EXEC(ICODE,SUBR2)
C
C---  LOAD AND EXECUTE THIRD SEGMENT
C
51    CALL EXEC(ICODE,SUBR3)
52    STOP
      END
C
      BLOCK DATA
      COMMON/IFLAG/IFLAG1
      END

FTN4,L
C
C
      PROGRAM SUBR1 (5), NAMED COMMON
```

```
C
      COMMON/IFLAG/IFLAG1
C
      WRITE(1,100)
100   FORMAT(/"SUBR1 HAS BEEN LOADED AND EXECUTED")
C
C---  SET IFLAG1 FOR ENTRY POINT IN PRGM1
C
      IFLAG1 = 1
      CALL PRGM1
      END

FTN4,L
C
C
      PROGRAM SUBR2 (5), NAMED COMMON
C
      COMMON/IFLAG/IFLAG1
C
      WRITE(1,100)
100   FORMAT(/"SUBR2 HAS BEEN LOADED AND EXECUTED")
C
C---  SET IFLAG1 FOR ENTRY POINT IN PRGM1
C
      IFLAG1 = 2
      CALL PRGM1
      END

FTN4,L
C
C
      PROGRAM SUBR3 (5), NAMED COMMON
C
      COMMON/IFLAG/IFLAG1
C
      WRITE(1,100)
100   FORMAT(/"SUBR3 HAS BEEN LOADED AND EXECUTED")
C
C---  SET IFLAG1 FOR ENTRY POINT IN PRGM1
C
      IFLAG1 = 3
      CALL PRGM1
      END
```

Vita

Wayne Todd was born on 1 January 1956 in Three Hills, Alberta, Canada to William M. Todd and H. Joyce (Hansen) Todd. He attended Meadowdale High School in Lynnwood, Washington and graduated in June 1974. In May of that year, he enlisted in the United States Air Force and wenton active duty in October. He was released from active duty in September 1976 under the Airman's Education and Commissioning Program to attend Oregon State University and participate in the AFROTC program there. In June 1980, he graduated with a Bachelor of Science Degree in Electrical Engineering. His next assignment was the Air Force Institute of Technology at Wright-Patterson AFB, Ohio, also in June 1980.

Permanent Address: 8516 Nesbit Ave. N.
Seattle, WA 98103

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/EE/82J-14	2. GOVT ACCESSION NO. AD-A118 066	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A GENERAL PURPOSE MINI-COMPUTER BASED DIGITAL SIGNAL PROCESSING LABORATORY.		5. TYPE OF REPORT & PERIOD COVERED MS Thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Wayne Todd 2Lt, USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN), Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE June, 1982
		13. NUMBER OF PAGES 174
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-17 23 JUL 1982

Lynn Wolaver

LYNN E. WOLAVER
Dean for Research and
Professional Development

AIR FORCE INSTITUTE OF TECHNOLOGY (ATC)
WRIGHT-PATTERSON AFB, OH 45433

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital Filters
Signal Processing
Digital Signal Processing
Fast Fourier Transform

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This project involved the integration of hardware and software to develop a general purpose digital signal processing (DSP) laboratory based on a Hewlett Packard (HP) 21MX-type computer. The hardware included an HP 2108A computer, HP 7906 disk drive, HP 2648A graphics terminal, Texas Instruments Silent 700 printing terminal, Remex optical paper tape reader, two Analog Devices ADC-12QZ analog-to-digital converters, and an HP 12555B digital-to-analog converter.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

The operating system generated for the laboratory was RTE-III. It is the latest version of RTE that can be used with the currently available hardware. Software support for I/O devices not supported by RTE-III was developed and implemented.

Four DSP applications programs were implemented and tested. The first program performs both a fast Fourier transform (FFT) and an inverse fast Fourier transform (IFFT) of a complex function. The FFT and IFFT are computationally efficient methods for time-to-frequency and frequency-to-time transformations, respectively. The second program designs a linear phase finite impulse response digital filter. The third program uses auto-correlation and covariance methods of linear prediction analysis. The fourth program designs a finite word-length infinite impulse response digital filter.

DAT

ILM